VŠB - Technická Univerzita Ostrava

Fakulta Strojní

Katedra mechaniky

Tvarová optimalizace ve strukturálních úlohách řešených pomocí MKP

Design Optimization in Structural Problems Solved by FEM

Student: Bc. Pavel Urubčík

Vedoucí diplomové práce : Ing. Jan Szweda, Ph. D.

Ostrava 2011

Prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě 23.5.2011 .....

.....

podpis studenta

Prohlašuji, že

- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§35 odst. 3).
- souhlasím s tím, že jeden výtisk diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o diplomové práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu §12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě : 23.5.2011

ANOTACE DIPLOMOVÉ PRÁCE

URUBČÍK, P. Tvarová optimalizace ve strukturálních úlohách řešených pomocí MKP. Ostrava: Katedra mechaniky, Fakulta strojní, Vysoká škola báňská – Technická univerzita Ostrava, 2011, 90 s. , vedoucí práce Ing. Jan Szweda, Ph. D.

Diplomová práce se zabývá tvarovou parametrickou optimalizací statických strukturálních úloh v kombinaci optimalizačního a výpočtového programu. Práce sleduje dva hlavní cíle. Testováním zjistit, zda-li existuje nějaká všeobecně aplikovatelná optimalizační metoda, která by byla vhodná pro řešení nejběžnějších lineárních statických strukturálních úloh v MKP, se kterými se konstruktér může setkat v praxi. Druhým cílem bylo zjistit efektivitu dostupných komerčních řešení parametrické optimalizace v porovnání s uživatelsky připraveným algoritmem v programu Matlab. Oba cíle byly dosaženy provedením optimalizace na třech testovacích úlohách s počtem parametrů 3-4. Analýza vybraných testovacích úloh ukázala, že ani v tak základní oblasti výpočtů, jako jsou statické strukturální úlohy, nelze generalizovat výběr vhodné optimalizační metody a že ani propracovaný optimalizační program nevyřeší všechna úskalí, se kterými se konstruktér může potkat.

ANNOTATION OF THESIS

URUBČÍK, P. Design Optimization in Structural Problems Solved by FEM. Ostrava: Department of Mechanics, Faculty of Mechanical Engineering, VŠB - Technical University of Ostrava, 2011, 90 p. , Thesis supervisor Ing. Jan Szweda, Ph. D.

The thesis deals with parametric shape optimization of static structural problems in combination of FEA software and pure mathematical optimization software. The aim of the thesis is twofold. Firstly, evaluation of which method is better for the specific range of problems in linear static structural analysis solved by FEM. The second aim is comparison of user written optimization methods in Matlab with optimization methods available in professional engineering optimization software. Both aims have been achieved by testing the methods on three testing problems with low number of parameters (3-4). Analysis of selected testing problems showed that it is not possible to find single method which could be efficiently used in solving these problems, not even in a basic range of problems such as linear static structural analysis. Besides, not even professional software can solve the problems that an engineer may encounter in this very basic are of optimization.

Obsah

Seznam použitých zkratek a symbolů....9
1.Úvod....10
1.1.Vymezení tématu práce....12
2.Teorie optimalizace....14
2.1.Postup....14
2.2.Optimalizační metody....15
2.3. Jedno-dimenzionální metody....16
2.3.1.Eliminační metody....17
2.3.2.Interpolační metody....21
2.4.Více-dimenzionální metody optimalizace....28
2.4.1.Metody využívající pouze funkční hodnoty....28
2.4.2.Metody vyžadující hodnotu 1.derivace....34
2.4.3.Metody vyžadující hodnotu 2.derivace....36
2.4.4.Stochastické metody....36
2.5.Shrnutí a volba metod pro řešení testovacích úloh....38
3.Rešerše optimalizačních programů....39
3.1.Průzkum trhu....39
3.2.OptiSLang [16]....41
3.2.1.Dostupné metody [8]....41
3.3.Matlab....44
3.4.ANSYS....45
3.5.Shrnutí a volba programů pro řešení testovacích úloh....46
4.Testování metod a optimalizačních programů....47
4.1.Metodika testování....47
4.1.1.Tvorba algoritmů – časová náročnost....48
4.1.2.Nastavení algoritmů....48
4.1.3.Metodologie vyhodnocení....48
4.2.Popis testovacích úloh a výsledky testování....49
4.2.1.Testovací úloha 1 – nosník konstantní pevnosti....50

4.2.2.Testovací úloha 2 – koncentrace napětí v nalisovaném spoji.....53
4.2.3.Testovací úloha 3 – omezení vrubového účinku na hřídeli.....57
4.2.4.Průzkum prostoru.....61
4.3.Celkové vyhodnocení testování.....65
5.Závěr.....67
6.Seznam použité literatury.....70
7.Seznamy tabulek a obrázků.....72
8.Přílohy.....75
8.1.Příloha 1 - Modely testovacích úloh a výpočty bez vrubu.....75
8.2.Příloha 2 - Průběhy objektivní funkce pro provedenou optimalizaci testovacích úloh.....77
8.2.1.Průběh iterací testovací úlohy č.1.....78
8.2.2.Průběh iterací testovací úlohy č.2.....82
8.2.3.Průběh iterací testovací úlohy č.3.....84
8.3.Příloha 3 – seznam přiložených souborů na CD.....89

Seznam použitých zkratek a symbolů

E	Youngův modul pružnosti [MPa]
EA	Evoluční algoritmus
Fobj	Hodnota objektivní funkce
Mk	Kroutící moment
S	Směrový vektor
TU1 (2,3)	Testovací úloha č.1 (2,3)
λ	Délka kroku (skalární veličina)
σ	Napětí [MPa]
μ	Poissonovo číslo [-]
ρ	Hustota [$t.mm^3$]

1. Úvod

V inženýrské praxi je žádoucí navrhnout konstrukci tak, aby její efektivita z hlediska požadavků účelovosti byla co možná maximální. V minulosti, kdy neexistovaly analytické metody ani výpočetní nástroje, bylo navrhování konstrukcí prováděno metodou pokusu a omylu. To bylo založeno pouze na zkušenosti a evoluci (konstrukčního návrhu). Metoda pokusu-omylu se využívá dodnes.

V dnešní době je však nežádoucí stále velmi rozšířený úkaz, kdy konstruktér s velkými výpočetními možnostmi užívá při návrhu konstrukce postupy z přelomu 19. a 20. století, kdy inženýři používali hlavně své intuice a invence. Proces návrhu v takových případech probíhá tak, že intuitivně navržená konstrukce je vzápětí náročně analyzována na výkonných počítačích s užitím moderních metod analýzy a drahými experimenty. Podle výsledku je potřeba pozměnit návrh a celý proces opakovat dokud se, víceméně náhodou, nenalezne nejvhodnější řešení. Toto řešení lze pak navíc stěží nazvat jako nejlepší dosažitelné.

Není však možné uvědomit si naprosto všechny souvislosti a skutečnosti a proto jsou empirické a evoluční metody stále užitečné, avšak pouze *jako doplněk* k racionálnímu přístupu navrhování konstrukcí, který nazýváme *konstrukční optimalizací*.

Konstrukční optimalizací se nazývá část procesu návrhu, kdy se v rámci daných možností hledá řešení pro dosažení maximální efektivity systému. Danými možnostmi jsou v užším slova smyslu myšleny návrhové parametry, které mohou zahrnovat vlastnosti materiálu, tvar konstrukce, rozměry a jiné charakteristiky mající vliv na podobu výsledného návrhu. Efektivita systému je pak dána účelovostí. Konstrukce musí jednak splňovat svoji hlavní funkci, pro kterou byla primárně navržena (tuhost konstrukce, hmotnost, posunutí, napětí) a zároveň také musí splňovat vedlejší podmínky (cena, výrobní čas, vyrobitelnost, geometrie).

Proces konstrukční optimalizace můžeme rozdělit do těchto fází [1]:

1. poznání vnějších podmínek
2. stanovení kritérií pro vyhodnocení optima
3. specifikace formy (topologie, tvar)
4. volba návrhových proměnných
5. stanovení vedlejších podmínek (probíhá současně s předchozími fázemi)

Tyto fáze lze shrnout jako *technická formulace konstrukčních cílů* a *sestavení matematického modelu fyzikálně zjednodušené technické úlohy.* Až po této formulaci nastupuje samotná optimalizace, kterou lze pro přehlednost také rozdělit do více fází:

6. volba vhodné matematické optimalizační metody
7. formalizace extremální úlohy
8. matematické řešení formalizované extremální úlohy
9. technická interpretace získaného řešení

Spojení procesu optimalizace a MKP analýzy

Každý návrh konstrukce je potřeba před použitím analyzovat, aby se zjistilo, zda-li vyhovuje účelu, pro který byl navržen. K tomu se v dnešní době s velkou oblibou používá *metoda konečných prvků* (MKP). S využitím MKP je možné levně a rychle provádět změny modelů, které předpovídají chování zkoumaných objektů ve skutečném světě. Toho se dá využít v optimalizaci při návrhu strojní součásti, kdy je potřeba provádět značný počet experimentů.

Poptávka po optimalizaci v kombinaci s MKP je evidentní z rozšiřování MKP programů o základní optimalizační nástroje, jak je možné pozorovat například v programu ANSYS Workbench. Na trhu jsou také specializované programy, které se zabývají pouze konstrukční optimalizací.

V případě hotových optimalizačních balíčků či kompletních programů je celý proces optimalizace integrován v jeden celek, což uživateli usnadňuje práci a zbavuje

ho nutnosti rozhodovat se nad méně důležitými prvky celého procesu. Pokud chce však konstruktér provádět optimalizaci a nemá k dispozici kompletní řešení, které spojuje MKP a optimalizaci, musí použít propojení matematického a MKP programu. Pokud navíc uživatel nechce nebo nemůže využít profesionální matematické řešení, musí si navrhnout celý proces sám. Na uživateli zůstává tedy i volba vhodné optimalizační metody, což vyžaduje znalost problematiky a praktické zkušenosti.

Existuje velké množství matematických optimalizačních metod, ale zdaleka ne všechny jsou vhodné pro konstrukční optimalizaci. Konstruktér musí vybrat metodu, která poskytuje dostatečně přesné výsledky v přípustném čase. Konstruktér může navíc použít pouze takovou metodu, ke které má dostatečné matematické a programové zázemí a která k řešení vyžaduje takové množství času a námahy kdy je řešení problému stále přínosné. Výběr vhodné metody je o to důležitější při optimalizaci v kombinaci s MKP, protože výpočty trvají mnohem déle než výpočty matematických funkcí.

V praxi se často v rámci jednoho celku (projektu, oddělení či firmy) řeší úlohy se stejnými charakteristikami. Mezi ty nejvíce používané patří statická analýza, přenos tepla, modální analýza či „buckling“. Drtivá většina dílů strojů a konstrukcí slouží primárně k přenosu sil. Lineární statická strukturální analýza se dá tedy považovat za základní kámen běžné konstrukční praxe a právě proto bych se v této práci chtěl zabývat optimalizací této skupiny úloh.

1.1. Vymezení tématu práce

Celý proces konstrukční optimalizace je široká oblast, ve které se lze zaměřit na kteroukoliv v úvodu zmíněných fází. V této práci bych se však chtěl zaměřit na fázi výběru vhodné optimalizační metody, jakožto části optimalizačního procesu, která je sice důležitá, ale na jejíž rozbor v praxi nebývá čas. Domnívám se, že v dnešní době dostupné výpočetní prostředky inženýrů rychle rostou, ale tyto kapacity nejsou při konstruování nových dílů náležitě využívány. Firmy, které vlastní MKP software tak vlastně již disponují prostředky, které by jim mohly přinést vyšší zisky kdyby tyto prostředky využívaly lépe v oblasti návrhu.

V práci se zaměřím na *parametrickou tvarovou optimalizaci lineárních*

strukturálních problémů, kam, jak se domnívám, spadá velké množství praktických výpočtů, protože spousta věcí se s dostatečnou přesností řešení dá zjednodušit právě na tento případ. Chtěl bych prozkoumat efektivitu metod pro aplikaci v této oblasti úloh a vybrat z nich tu nejvhodnější, pokud existuje. Do práce zahrnu také malý průzkum dostupných optimalizačních komerčních řešení a porovnám tento software s uživatelem vytvořenými algoritmy. Bude se tedy jednat o zkoumání metod matematické optimalizace v propojení s MKP programem, který slouží pouze pro vyhodnocení úlohy v závislosti na vstupních proměnných.

Očekávaným přínosem je zkušenost s optimalizačními metodami ve specifické oblasti strukturálních úloh, ujasnění efektivity algoritmů na praktických příkladech a zjištění za jakých podmínek je vhodnější koupit hotový produkt, vyrobit si vlastní nástroj nebo zda-li se do optimalizace vůbec pouštět. Optimalizování součásti sice šetří prostředky, ale jen tehdy když výdaje s ní spojené jsou nižší než ušetřené náklady.

2. Teorie optimalizace

2.1. Postup

Již bylo zmíněno, že proces optimalizace konstrukce začíná návrhem konstrukce a až po té přichází na řadu samotný optimalizační proces. Nyní budou podrobněji popsány jednotlivé fáze:

Návrh konstrukce

1. **Poznání vnějších podmínek**
   Podmínky provozování konstrukce.
2. **Stanovení kritérií pro vyhodnocení optima**
   Činitelé pro hodnocení účelu konstrukce.
3. **Specifikace formy (topologie, tvar)**
   Vzájemné uspořádání prvků konstrukce.
4. **Volba návrhových proměnných**
   Bývá analyticky nejobtížnějším krokem, protože volba parametrů předurčuje úspěch optimalizace.
   Tvarové proměnné – určují tvarové uspořádání návrhu konstrukce. Rozměrové proměnné – definují konkrétní rozměry, např. průměr, tloušťka. Materiálové proměnné – určují mechanické vlastnosti konstrukce.
5. **Stanovení vedlejších (omezujících) podmínek**
   Představují další požadavky kladené na hledané optimální řešení. Určují přijatelnost určitého konkrétního návrhu. Typicky: max. napětí, min. frekvence, omezení rozměrů.
   Do řešení obvykle vstupují v podobě nerovnic.

Optimalizace parametrů konstrukce

6. **Volba vhodné matematické optimalizační metody**
   Výběr dostupné metody podle charakteru úlohy.

7. **Formalizace extremální úlohy**

   Sestavení cílové funkce f(x).

   Formulace úlohy optimalizace min f(x) či max f(x) (což je vlastně max -**f**(x)).

   Vymezení prostoru přípustných řešení – zahrnutí omezujících podmínek.

8. **Matematické řešení formalizované extremální úlohy**

   Samotné řešení (úspěch, rychlost, přesnost) výrazně závisí na charakteru úlohy a volbě optimalizační metody.

9. **Technická interpretace získaného řešení**

   „Přeložení“ výsledků do jazyka technické praxe. Úprava výsledků pro praktické uplatnění (výběr normovaných polotovarů, vyhlazení obrysů).

2.2. Optimalizační metody

Po formulaci úlohy (viz předcházející postup body 1 až 7) přichází na řadu matematická optimalizace. Cílem je nalezení nejlepšího hodnot návrhových proměnných z hlediska vyhodnocovaných kritérií.

V následujících odstavcích budou popsány používané metody numerické optimalizace v technické praxi. Ty lze dělit do skupin podle různých charakteristik jednotlivých metod:

Dle řádu úlohy

1. Metody využívající pouze funkční hodnoty: přímé hledání, Simplex metoda.
2. Metody vyžadující hodnotu 1.derivace: gradientní metody *Hooke-Jeeves*, *kvazi-Newtonova metoda*, *Cauchyho metoda*.
3. Metody vyžadující hodnotu 2.derivace: Newtonova metoda.

Heuristické metody

4. Evoluční algoritmy: Simplex, evoluční algoritmy, simulované žíhání („simulated annealing“)
5. Stochastické metody: Roj částic („particle swarm“)

Existuje velké množství jiných metod (např. metoda neuronových sítí). V technické praxi se však využívají výše uvedené skupiny, především pak první čtyři [9].

2.3. Jedno-dimenzionální metody

Princip jedno-dimenzionálního hledání funkce více proměnných

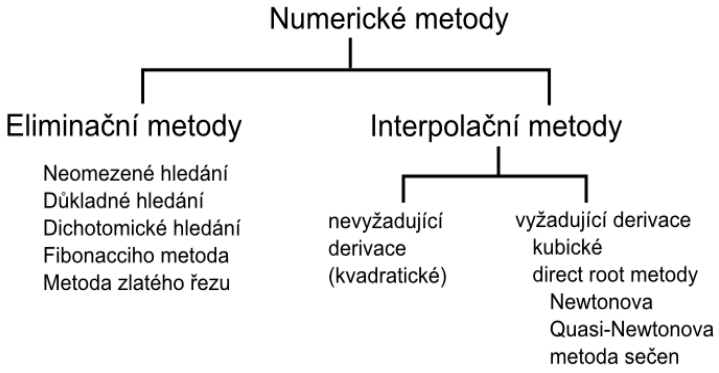
V praxi zřídka narazíme na optimalizaci funkce jedné proměnné. Metody jedno-dimenzionální optimalizace mají však uplatnění u některých metod více-dimenzionální optimalizace. Více-dimenzionální ($X = \{x_1, x_2, .., x_n\}$) optimalizační úlohu lze zapsat a řešit ve tvaru

$$X_{i+1} = X_i + \lambda^* \cdot S \tag{2.1}$$

kde optimální délka kroku je **λ*** a **S** je známý směr hledání jednotlivých parametrů. Řešení úlohy se omezí na nalezení řešení

$$\lambda^* = \min_{\lambda_i} [f(X_i + \lambda_i \cdot S_i)] \tag{2.2}$$

Jedno-dimenzionální optimalizační metody lze rozdělit na eliminační a interpolační (viz schéma na obr.1). Na následujících stránkách budou uvedeny, shrnuty a vyhodnoceny optimalizační algoritmy (metody) obou skupin. Detailnější popis bude však uveden pouze u těch, které jsou pro danou problematiku prokazatelně efektivnější.

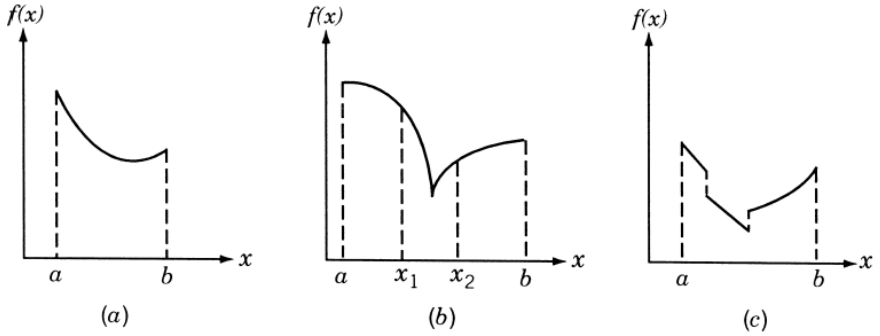


Obrázek 2.1: Dělení jedno-dimenzionálních optimalizačních metod [4]

Unimodální funkce

Unimodální funkce je taková funkce, která má pouze jeden vrchol (nebo údolí) na daném intervalu. Funkce se nazývá unimodální pokud z dvou bodů na stejné straně od optima má ten bod nižší funkční hodnotu, který je blíže optimu.

Všechny eliminační metody předpokládají unimodalitu. Pokud je funkce multimodální tak je nutné rozdělit stávající interval na více částí.



Obrázek 2.2: ukázka unimodálních funkcí [4]

2.3.1. Eliminační metody

Optimalizační metody, které hledají optimum funkce jedné proměnné postupným zmenšováním intervalu nejistoty (interval na kterém se nachází hledané optimum) se nazývají *eliminační metody*. Tyto metody zpravidla předpokládají unimodalitu funkce na intervalu kde je optimum hledáno. Unimodalita totiž zajišťuje to, že stačí porovnat funkční hodnoty ve dvou různých bodech intervalu k tomu, abychom předpověděli kterému z nich je bod optima blíže.

Existuje několik základních přístupů, které se používají při hledání funkce jedné proměnné.

Neomezené hledání („unrestricted search")

V praktických úlohách bývá obvykle vymezená oblast hledání, ale může se stát, že potřebujeme najít optimum na neomezeném intervalu. Základní přístupy jsou:

1. hledání s neměnným krokem – z počátečního bodu se začne hledat v příznivém

směru s pevným krokem. Pro určení optima se porovnávají dva poslední body. Krok musí být dostatečně malý pro dosažení požadované přesnosti. Je dobře implementovatelný avšak v mnoha případech je neefektivní.

2. hledání s urychlujícím krokem – podobné předchozí metodě, ale krok se zvětšuje s každým novým bodem (např. zdvojnásobí předchozí interval). Metoda může být nadále vylepšena tím, že při nalezení intervalu s optimem se bude tento prohledávat důkladněji se zmenšeným krokem.

Úplné prohledávání („exhaustive search")

Jako všechny následující metody i tato slouží k hledání optima na vymezeném intervalu. Tato metoda prohledá prostor intervalu s předem určeným počtem bodů, čímž zredukuje interval nejistoty. Metoda patří do skupiny *metod simultánního hledání*. Vyžaduje vyhodnocení všech předem určených bodů a je relativně neefektivní v porovnání s *metodami sekvenčního hledání*.

Dichotomické hledání

V každé iteraci jsou v těsné blízkosti středu intervalu nejistoty vyhodnoceny dva body, čímž se interval nejistoty zmenší o téměř polovinu.

Metoda půlení intervalu (interval halving)

Výkonnostně podobná dichotomickému hledání. Na začátku jsou vyhodnoceny tři místa (kraje intervalu a střed) a v každém dalším kroku dvě nová místa, podle nichž se určuje která půlka předchozího intervalu bude vyloučena v které bude hledání pokračovat.

Fibonacciho metoda

Je efektivní metoda, která však na začátku hledání musí mít určen počet vyhodnocení funkce (n). Využívá Fibonacciho čísel. F(n): 1, 1, 2, 3, 5, 8, 13, ..

(součtem dvou posledních se získá následující). Na počátku experimentu se určí počáteční krok L_2^* (dle 2.3) a vyhodnotí se body ve vzdálenosti L_2^* od obou hranic intervalu L_0. Následně se interval zkracuje za předpokladu *unimodality* funkce. Z Fibonacciho čísel se pro určení zkrácení intervalu nejistoty využívá jen jedno číslo (dle zvoleného ***n***) a každé další zkrácení vychází z tohoto čísla a předchozích bodů. Důležitý je fakt, že každé další zkrácení vyžaduje pouze jedno vyhodnocení funkce.

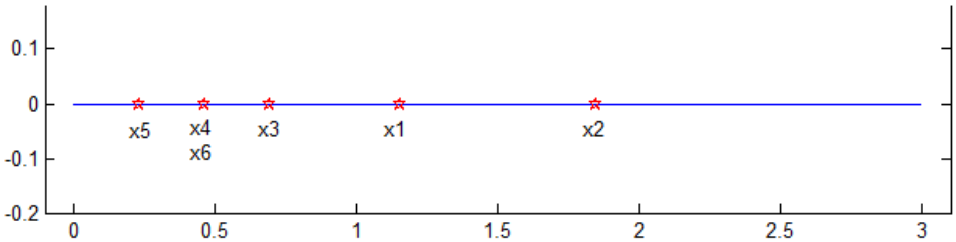
$$L_2^* = \frac{F_{n-2}}{F_n} \cdot L_0 \tag{2.3}$$

Příklad: Hledáme minimum funkce f (x) na intervalu <0, 3> Fibonacciho metodou s počtem vyhodnocení funkce n = 6. L_0 je tedy 3.

$$L_2^* = \frac{5}{13} \cdot 3 = 1{,}153846 \tag{2.3*}$$

První dva experimenty budou tedy na $x_1 = 1{,}153846$; $x_2 = 3 - 1{,}153846 = 1{,}846154$. Předpokládejme $f_1 = f(x_1) < f_2 = f(x_2)$. Za použití předpokladu *unimodality* funkce nebudeme dále uvažovat interval $[x_2, 3]$. Následující $x_3 = 0 + (x_2 - x_1) = 1{,}846154 - 1{,}153846 = 0{,}692308$. V tomto případě předpokládejme $f_1 > f_3$ a proto dále neuvažuje interval $[x_1, x_2]$. Postup se opakuje: $x_4 = 0 + (x_1 - x_3) = 0{,}461538$. Hodnota $f_4 < f_3$: neuvažuje tedy interval $[x_3, x_1]$ a pokračujeme až do x_6 kdy zjistíme, že poměrný interval nejistoty je $L_6/L_0 = 0{,}076923$.

Toto číslo není náhodné, protože u Fibonacciho m. je vždy konečný poměrný interval nejistoty roven $1/F_n$. Aplikováno na uvedený příklad: $1/F_6 = 1/13 = 0{,}076923$.



Obrázek 2.3: umístění bodů x_1-x_6 v iteračním procesu při Fibonacciho hledání n = 6 na intervalu <0, 3>

Metoda Zlatého řezu ("Golden section method")

Metoda je podobná Fibonacciho metodě. Poměr, nazývaný Zlatý řez, o který se v každé iteraci interval nejistoty zkrátí je téměř shodný s intervalem zkrácení z předchozí metody, ale liší se tím, že konečný počet iterací není dán předem.

Konstanta Zlatého řezu pro určení poměru zkrácení intervalu v každé iteraci je kladná část kořene rovnice $\gamma^2+\gamma-1=0$ a činí přibližně 0,618. Postup zmenšování intervalu je také téměř shodný s Fibonacciho metodou a nebude zde dále popsán[1].

Metoda Zlatého řezu je použita při výpočtu *Cauchyho metodou* a je uložena v souboru *ZlatyRezExpUru.m* na přiloženém disku.

Vyhodnocení efektivity eliminačních metod

Eliminační metody funkce jedné proměnné lze porovnávat podle poměru zkrácení intervalu nejistoty v každém kroku. Z následující tabulky vyplývá, že daleko nejlépe v tomto ohledu jsou na tom Fibonacciho metoda a m. Zlatého řezu. Obě metody fungují na stejném principu zkracování, ale liší se v definici toho kolik iterací bude provedeno.

Metoda	Rovnice intervalu	n = 5	n = 10
Úplné prohledávání	$L_n=\frac{2}{n+1}L_0$	0,333 L_0	0,1818 L_0
Dichotomické prohledávání	$L_n=\frac{L_0}{2^{n/2}}\delta\left(1-\frac{1}{2^{n/2}}\right)$	L_0/4+0,0075 pro n = 4 L_0/8+0,00875 pro n = 6	0,03125 L_0 + 0,009687
Půlení intervalu	$L_n=\left(\frac{1}{2}\right)^{(n-1)/2}L_0$	0,25 L_0	0,0625 L_0 pro n = 9 L_0/8+0,00875 pro n = 11
Fibonacciho metoda	$L_n=\frac{1}{F_n}L_0$	0,125 L_0	0,01124 L_0
Metoda Zlatého řezu	$L_n=(0{,}618)^{n-1}L_0$	0,1459 L_0	0,01315 L_0

Tabulka 2.1: porovnání eliminačních metod [4]

1 Více o původu Zlatého řezu, odvození vztahu a aplikaci lze najít např. V [3, s.51] nebo [4 s.270].

2.3.2. Interpolační metody

Nevýhodou *eliminačních metod* je, že neuvažují velikost změny při vyhodnocení, ale pouze porovnávají hodnotu objektivní funkce. Této hodnotné informace však využívají metody interpolační (někdy také nazývané *polynomické aproximační metody*). Interpolační metody byly původně navrženy jako jedno-dimenzionální hledání pro funkci více proměnných a jsou v tomto ohledu také obecně považovány za efektivnější než eliminační metody.

Jak bylo vysvětleno výše úlohy více-dimenzionální optimalizace lze převést na jedno-dimenzionálního hledání. Pokud by byl znám explicitní tvar řešené funkce, cílem jedno-dimenzionální optimalizace pro funkci více proměnných by se omezilo na nalezení **λ***, tj. takové nezáporné hodnoty **λ**, při které dosáhne funkce

$$f(\lambda) = f(X + \lambda S) \tag{2.4}$$

lokálního minima. Původní úloha f (X), jakožto explicitní funkce x_i (i = 1, 2 ,.., n), je tedy nahrazena úlohou f (λ). Řešení lze získat vyřešením rovnice

$$\frac{df}{d\lambda}(\lambda) = 0 \tag{2.5}$$

k nalezení λ* při daných vektorech X a S.

Ukázka odvození jedno-dimenzionální úlohy z více-dimenzionální:

Minimalizovat $f(X) = x_1 - x_2^2$

pro startovní bod $X_1 = \begin{Bmatrix} 2 \\ 2 \end{Bmatrix}$ *ve směru hledání* $S = \begin{Bmatrix} -1 \\ -1 \end{Bmatrix}$

Nový bod X *pak může být vyjádřen jako*:

$$X = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = X_1 + \lambda S = \begin{Bmatrix} 2-\lambda \\ 2-\lambda \end{Bmatrix}$$

Tímto nahrazením získáme f *jako funkci* λ:

$$f(\lambda) = \begin{Bmatrix} 2-\lambda \\ 2-\lambda \end{Bmatrix} = 2 - \lambda - (4 + 4\cdot + \lambda^2) = -\lambda^2 + 3\cdot\lambda - 2$$

Minimalizací této funkce získáme **λ*** při které je v daném směru hodnota funkce nejnižší.

V inženýrských problémech řešených pomocí MKP však explicitní tvar funkce neznáme. Interpolační metody slouží právě k tomu, aby nalezly tvar explicitní funkce (například ve tvaru polynomu), která vystihuje průběh objektivní funkce při řešení skutečného problému.

Nezbytným předpokladem pro použití těchto aproximačních přístupů je dostatečná hladkost skutečné funkce. Metodika by nebyla efektivní u nelineárních problémů (např. s diskrétními parametry).

Kvadratická interpolace

Metoda kvadratické interpolace, stejně jako eliminační metody, nevyžaduje derivaci funkce, ale využívá pouze funkční hodnotu.

Postup nalezení takového kroku **λ***, který určuje hodnotu parametrů s nejmenší funkční hodnotou dle rovnice (2.1) je ve třech krocích.

1. Normalizování vektoru **S**.
2. Aproximace původní funkce **f (λ)** kvadratickou funkcí **h (λ)** a nalezení jejího minima $\tilde{\lambda}$*****.
3. Pokud $\tilde{\lambda}$***** není dostatečně blízko skutečnému minimu **λ***, je vytvořena nová („opravená“) kvadratická funkce a proces body 1, a 2, jsou zopakovány.

Krok 1 – Pokud je úloha více-dimenzionální vektor proměnných **S** se normalizuje tak aby krok **λ** = 1 byl přijatelný. Obvykle se každá hodnota vektoru **S** podělí **Δ**. To se určí buď jako **Δ = max|s$_i$|** kde **s$_i$** značí jednotlivé komponenty **S**. Druhý způsob je, **Δ = (s$_1^2$ + s$_1^2$ + .. + s$_n^2$)$^{1/2}$**.

Krok 2 – Nechť

$$h(\lambda)=a+b\lambda+c\lambda^2 \tag{2.6}$$

je kvadratickou funkcí, která je aproximací **f (λ)**. Nezbytná podmínka minima

h (λ) je

$$\frac{dh}{d\lambda} = b + 2c\lambda = 0 \tag{2.7}$$

Pro zjištění konstant ***a, b, c*** je potřeba vyhodnotit funkci ve třech bodech **A**, **B** a **C**.

$$\begin{aligned} fa &= a + b\,A + c\,A^2 \\ fb &= a + b\,B + c\,B^2 \\ fa &= a + b\,C + c\,C^2 \end{aligned} \tag{2.8}$$

Řešení rovnice (2.8) je následující

$$a = \frac{fa\,B\,C(C-B) + fb\,C\,A(A-C) + fc\,A\,B(B-A)}{(A-B)(B-C)(C-A)} \tag{2.9}$$

$$b = \frac{fa(B^2-C^2) + fb(C^2-A^2) + fc(A^2-B^2)}{(A-B)(B-C)(C-A)} \tag{2.10}$$

$$c = \frac{-fa(B-C) + fb(C-A) + fc(A-B)}{(A-B)(B-C)(C-A)} \tag{2.11}$$

a minimum $\tilde{\lambda}^*$ funkce **h (λ)** plyne z rovnic (2.7), (2.10) a (2.11)

$$\tilde{\lambda}^* = \frac{-b}{2c} \tag{2.12}$$

Krok 3 – $\tilde{\lambda}^*$ nalezené v kroku 2 je minimum **h (λ)**. V tomto kroku se zjišťuje, zda-li je $\tilde{\lambda}^*$ dostatečně blízko skutečnému minimu **λ*** funkce **f (λ)** (v rámci kritéria přesnosti nastavené uživatelem). Výpočetně méně náročným způsobem je vyhodnocení poměrného rozdílu hodnot **h ($\tilde{\lambda}^*$)** a **f (λ)**

$$\left| \frac{h(\tilde{\lambda}^*) - f(\tilde{\lambda}^*)}{f(\tilde{\lambda}^*)} \right| \leqslant \varepsilon_1 \tag{2.13}$$

Je také možné ověřit, zda-li je v okolí $\tilde{\lambda}^*$ hodnota **df/dλ** blízko nule, tj. funkce v okolí bodu příliš nemění svou hodnotu. V MKP derivaci neznáme a pomůžeme si numericky

$$\left| \frac{f(\tilde{\lambda}^* + \Delta\tilde{\lambda}^*) - f(\tilde{\lambda}^* - \Delta\tilde{\lambda}^*)}{2\Delta\tilde{\lambda}^*} \right| \leqslant \varepsilon_2 \tag{2.14}$$

Pokud není kritérium (ε_1 nebo ε_2) splněno, použije se opět rovnice (2.6) a hledá se nová vhodnější interpolace **f (λ)**. Nové konstanty ***a, b*** a ***c*** pro interpolovanou kvadratickou křivku se získají interpolací ve třech nejlepších doposud vypočtených bodech.

Kvadratická interpolace je použita při výpočtu *Cauchyho metodou* a je uložena v souboru *quadint.m* na přiloženém disku.

Kubická interpolace

Metoda hledá minimum funkce **f (λ)** nalezením minima náhradní kubické funkce **h (λ)**. Využívá u toho derivaci:

$$f'(\lambda)=\frac{d f}{d \lambda} f(X+\lambda S)=S^T \nabla f(X+\lambda S) \tag{2.15}$$

Proces má 4 fáze podobné kvadratické interpolaci. Ve stručnosti:

Krok 1. Normalizace směrového vektoru **S**.

Krok 2. Určení intervalu na kterém leží minimum. Tj. Hledají se takové dva body, jejichž sklon ***df/fλ*** ,má odlišné znaménka.

Krok 3. Nalezení minima $\tilde{\lambda}$* funkce.

$$h(\lambda)=a+b\lambda+c\lambda^2+\lambda^3 \tag{2.16}$$

Využívá se u toho derivací v krajních bodech **A** a **B**.

Krok 4. Ověřuje se, zda-li je $\tilde{\lambda}$* dostatečně blízko skutečnému minimu **f (λ)**.

Metody přímého hledání (Direct root methods)

Nezbytná podmínka v bodě minima funkce **f (λ)** je ***f '(λ*)*** = *0*. Skupina metod přímého hledání kořene je přímo zaměřena na nalezení ***λ**** tímto způsobem. Patří sem *Newtonova metoda*, *kvazi-Newtonova metoda* a *metoda sečen*.

Newtonova metoda

Uvažujme (2.17) jako kvadratickou aproximaci funkce **f (λ)** za použití Taylorova rozvoje

$$f(\lambda)=f(\lambda_i)+f'(\lambda_i)(\lambda-\lambda_i)+\frac{1}{2}f''(\lambda_i)(\lambda-\lambda_i)^2 \tag{2.17}$$

Derivací (2.17) se získá

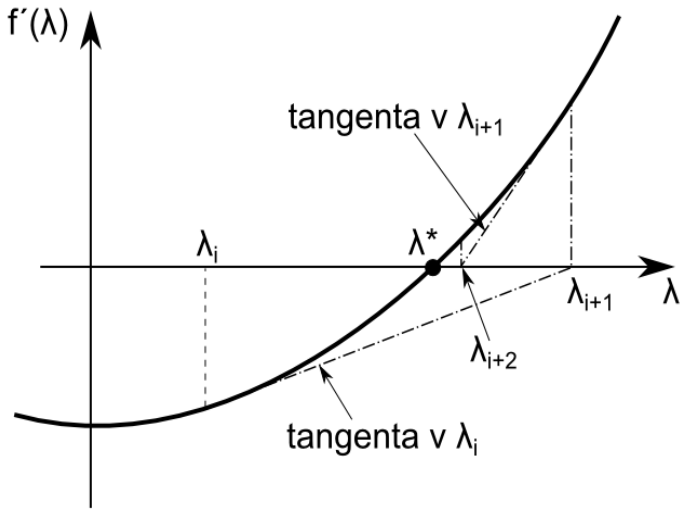
$$f'(\lambda)=f'(\lambda_i)+f''(\lambda_i)(\lambda-\lambda_i) \tag{2.18}$$

Odkud, položíme-li rovnici rovnu nule, získáme aproximaci minima λ_{i+1} jako funkci předchozího bodu a jeho derivací:

$$\lambda_{i+1}=\lambda_i-\frac{f'(\lambda_i)}{f''(\lambda_i)} \tag{2.19}$$

Pro zjištění zda-li je λ_{i+1} dostatečně blízko minima se kontroluje $|f'(\lambda_{i+1})| \leq \varepsilon$, kde ε je malé číslo vyjadřující požadovanou přesnost.

Newtonova metoda konverguje velice rychle pokud pro funkci platí $f''(\lambda_i) \neq 0$. S ohledem na kombinaci s MKP výpočty ji však nelze přímo použít, kvůli nedostupnosti derivací. Pro tyto účely slouží z ní odvozená následující metoda.



Obrázek 2.4: průběh iterací u Newtonovy metody

Kvazi-Newtonova metoda

U MKP úloh nejsou dostupné derivace, které jsou nezbytné pro použití *Newtonovy*

metody. Derivace v rovnici (2.19) však lze aproximovat následovně:

$$f'(\lambda_i) = \frac{f(\lambda_i + \Delta\lambda) - f(\lambda_i - \Delta\lambda)}{2\Delta\lambda} \tag{2.20}$$

$$f''(\lambda_i) = \frac{f(\lambda_i + \Delta\lambda) - 2f(\lambda_i) + f(\lambda_i - \Delta\lambda)}{\Delta\lambda^2} \tag{2.21}$$

kde $\Delta\lambda$ je velmi malý krok.

Takto upravená rovnice (2.19) se nazývá *kvazi-Newtonova metoda*. Kontrola konvergence iteračního procesu se opět provádí jako $|\mathbf{f}'(\lambda_{i+1})| \leq \varepsilon$.

Z výše uvedeného plyne, že tato metoda vyžaduje v každé iteraci ještě vyhodnocení funkce ve dvou bodech: $(\lambda_i + \Delta\lambda)$ a $(\lambda_i - \Delta\lambda)$.

Metoda sečen

Metoda sečen využívá podobnou rovnici jako Newtonova metoda (2.18):

$$f'(\lambda) = f'(\lambda_i) + s(\lambda - \lambda_i) = 0 \tag{2.22}$$

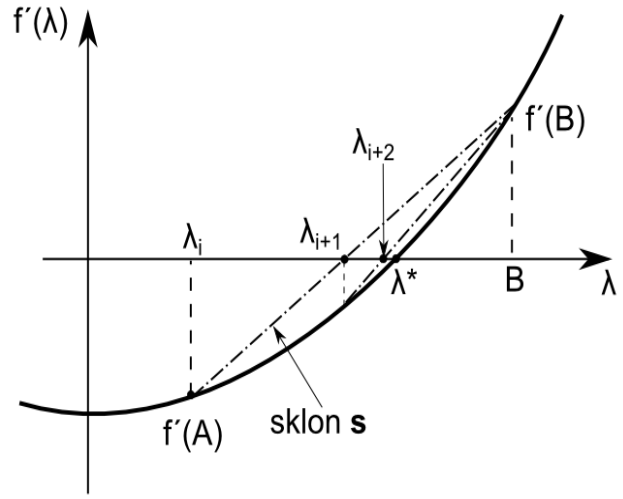
kde **s** je směr určující sklon křivky spojující derivace funkce v bodech **A** a **B** tak jak je znázorněno na obrázku. Sklon **s** lze vyjádřit jako

$$s = \frac{f'(B) + f'(A)}{B - A} = 0 \tag{2.23}$$

Jak je patrné z obrázku 2.5, metoda aproximuje minimum (tj.$\mathbf{f}'(\lambda) = 0$) jako lineární rovnici. Každý následující aproximace minima bude

$$\lambda_{i+1} = \lambda_i - \frac{f'(\lambda)}{s} = A - (B - A)\frac{f'(A)}{f'(B) - f'(A)} \tag{2.24}$$

Na metodu sečen se dá nahlížet jako na kombinaci eliminační a Newtonovy metody.



Obrázek 2.5: Průběh iterací metody sečen, znázornění parametru ***s***

Porovnání metod jedno-dimenzionální optimalizace

Ze skupiny eliminačních metod je, s ohledem na délku zkrácení intervalu, nejefektivnější *Fibonacciho metoda* spolu s *metodou Zlatého Řezu*. Eliminační metody jsou efektivnější pokud je definován interval nejistoty. Pokud není definován, vhodnější jsou *kvadratická interpolace* a *kvazi-Newtonova metoda*. Další metody jako *kubická interpolace* či *metoda sečen* jsou ještě efektivnější., vyžadují však znalost derivace zkoumané funkce, což u numerických výpočtů neznáme a lze jej pouze aproximovat přídavným výpočtem v okolí zkoumaného bodu.

Pokud tedy více-dimenzionální metody budou pro svou funkcionalitu využívat jedno-dimenzionální metody, bude použita *metoda Zlatého řezu* (upřednostněna před *Fibonacciho m.* z důvodu jednodušší aplikace) a *kvadratická interpolace*. Vzhledem k tomu, že v úlohách je často nutné definovat intervaly pro hodnoty parametrů, musí být tyto metody před použitím ošetřeny.

2.4. Více-dimenzionální metody optimalizace

2.4.1. Metody využívající pouze funkční hodnoty

Jedná se o metody 0.řádu, nazývané též *metodami přímého hledání* či ne*gradientními metodami*. Jsou vhodnější pro řešení jednodušších úloh s nízkým počtem proměnných. Obecně jsou méně efektivnější než gradientní metody [9].

Metoda přímého hledání („Direct Search Method“)

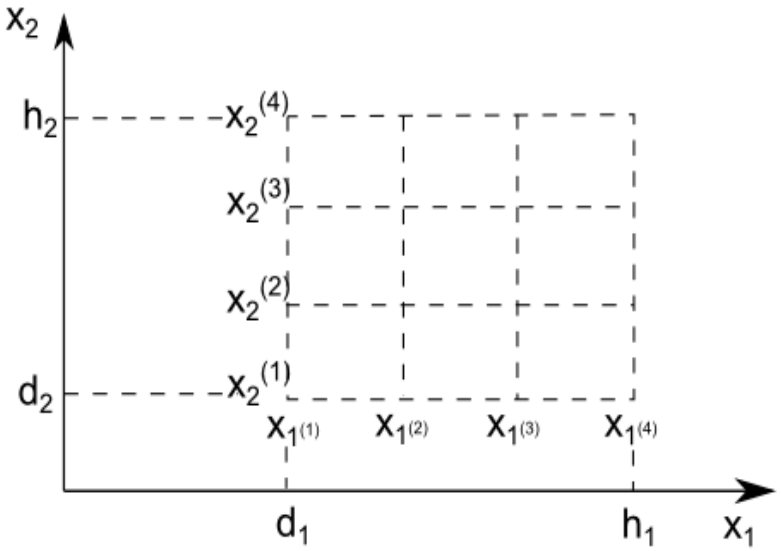
Tato skupina metod využívá náhodně vygenerovaných čísel k prohledání prostoru. Existuje několik přístupů jakými se náhodně vygenerovaná čísla používají:

1. Metoda náhodného skoku („Random Jumping M.“)– prohledává celý prostor v rámci daného intervalu. Generovaná hodnota proměnné v každém směru je $\mathbf{x_i} = \mathbf{x_{imin}} + \mathbf{r_i}\ (\mathbf{x_{imax}} - \mathbf{x_{imin}})$, kde $\mathbf{x_{imin}}$ a $\mathbf{x_{imax}}$ jsou hranice intervalu a $\mathbf{r_i}$ je náhodné číslo na intervalu <0; 1>.
2. Metoda náhodného kroku („Random Walk M.“) – je založená na postupu generování vylepšených aproximací. Každý krok zohledňuje výsledek předchozího odhadu a je založen na rovnici (2.1).
3. Metoda náhodného kroku s využitím nalezeného směru – je vylepšení předchozí metody s tím, že pokud je náhodným generováním nalezen směr ve kterém hodnota funkce klesá, je využití tohoto směru dále znásobeno větší délkou kroku.

Výhodou této metody je možnost prohledávání funkce, která není spojitá ve všech bodech. Má využití spíše při globálním prohledávání funkce, kde se očekává více lokálních minim. Využívají se tam, kde ostatní metody selhávají. Tyto funkce vyžadují vysoký počet iterací, avšak mohou být použity v rané fázi optimalizace a následně nahrazeny efektivnějšími metodami k nalezení minima.

Prohledávání v síti („Grid Search M.“)

Jedná se o rovnoměrné prohledání prostoru proměnných. V daných mezích se každý směr rozdělí na určitý počet míst ve kterých se pak vyhodnocuje optimalizovaná funkce. Tento přístup vyžaduje vysoký počet vyhodnocení funkce. Pokud by například měl tímto způsobem být prohledán prostor s 10 proměnnými a v každém směru by byly vyhodnoceny pouze 3 místa ,bylo by potřeba 3^{10} = 59 049 funkčních vyhodnocení. Při nižším počtu proměnných však může sloužit k přibližnému nalezení globálního minima stejně jako předchozí metoda.



Obrázek 2.6: Metoda prohledávání v síti

Metoda cyklické záměny parametrů

Někdy také nazývána *Gauss-Seidelova metoda* [9], nebo v angličtině „*Univariate Method*“ [4]. Minimum funkce se hledá postupně v jednotlivých směrech tak, že se mění pouze jeden parametr a ostatní zůstávají ve stejné iteraci nezměněny. Po té co je v daném směru nalezeno minimum se obměňuje jiný parametr a postup se opakuje až do vystřídání všech směrů. Tato sekvence se opakuje až do uspokojivého nalezení optima funkce.

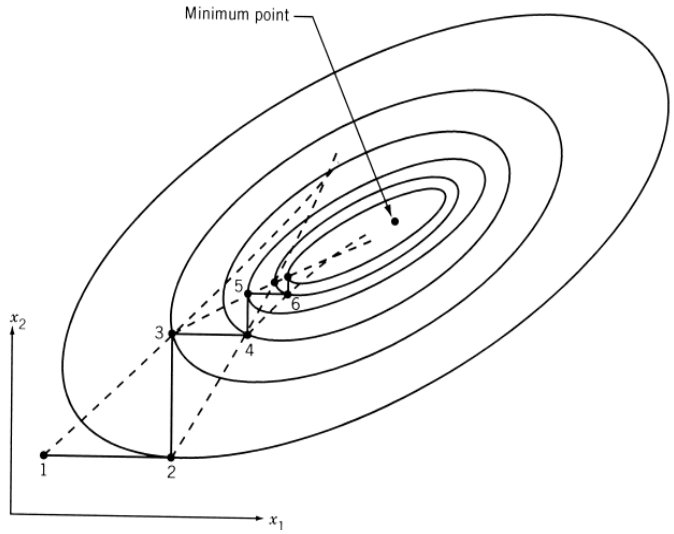
Úloha se tímto postupem stává úlohou jedno-dimenzionální optimalizace a řeší se tedy metodami zmíněnými v kapitole 2.3.

Metoda je jednoduše aplikovatelná, avšak konverguje pomalu, především pak v

blízkém okolí minima, kdy má metoda tendenci oscilovat což značně navyšuje počet iterací. Někdy vzniká také *problém ostrých hran* při kterém dochází k zacyklení [9]. V 2-dimenzionální úloze si lze tuto oscilaci představit jako zubatou přímku která směřuje k optimu a jejíž zuby se zjemňují s tím jak se proměnné blíží přesnému řešení. Z tohoto důvodu je u této metody vhodné iterační proces včas zastavit. V případě funkce ve tvaru strmého údolí tato metoda ani nemusí vůbec konvergovat.

Metoda předlohového hledání („Pattern search method“)

V předchozí metodě byl prostor prohledáván ve směrech os jednotlivých parametrů. To, jak bylo uvedeno, vede k problémům s konvergencí v určitých případech. Právě *Metoda předlohového hledání* se snaží tento problém eliminovat a to tak, že se vhodně upraví směr prohledávání ve směru poklesu funkční hodnoty. Následující obrázek funkce dvou proměnných znázorňuje jak po každé sekvenci hledání (body 1,3; 2,4; atd.) směřují čáry procházející těmito body k minimu. Tyto směrnice, které slouží jako předloha dalšího směru hledání jsou základní myšlenkou metody.



Obrázek 2.7: Metoda předlohového hledání. Přerušované čáry vyznačují řídící směry pro další iterace, které metoda využívá.[4]

Jednodušší metoda postavená na tomto principu je metoda *Hooke-Jeeves* [9], která určuje nový směr z posledních dvou sekvencí a následně v daném směru akceleruje.

Nejznámějším a také nejefektivnějším zástupcem této skupiny metod je však Powellova metoda.

Powellova metoda

Tuto nejrozšířenější metodu ze skupiny metod přímého hledání lze také zařadit do skupiny *metod sdružených směrů*. *Metody sdružených směrů* mají vlastnost minimalizovat kvadratickou funkci v konečném počtu kroků, čímž urychlují konvergenci i obecně nelineární funkce.

Powellova metoda je specifická způsobem jakým vybírá směr pro následující iteraci (viz obr.2.8). Ten se opírá o dvě vlastnosti směrových vektorů:

1. Směrové vektory určující další bod pro vyhodnocení funkce mají sdružené směry. Směrové vektory $\mathbf{S_i}$ a $\mathbf{S_j}$ se nazývají sdružené (konjugované) k **A** pokud platí, že

$$S_i^T A S_j = 0 \quad pro všechna \quad i \neq j, \quad kde \quad i, j = 1,2,..,n \tag{2.25}$$

Tato sdruženost také zaručuje lineární nezávislost obou vektorů. Stejná rovnice platí pro určování ortogonálních směrů, pouze matice **A** není libovolná matice (která je právě tou společnou charakteristikou vektorů $\mathbf{S_i}$ a $\mathbf{S_j}$) ale A by v takovém případě byla matice jednotková.

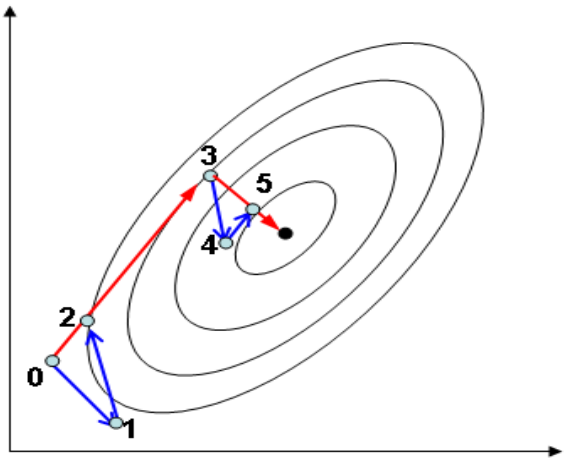
2. Nový směr v každé iteraci se určí jako kombinace předchozích směrů. (detailněji v [3], sekce 3.2.3)

Bylo prokázáno[2], že Powellova metoda patří do skupiny *metod sdružených směrů* a má proto teoreticky kvadratickou konvergenci (tzn. nalezení minima v *n* krocích kvadratické funkce o *n* proměnných). V praktických úlohách je však počet iterací o něco větší z důvodu aproximace délky kroku.

Powellova metoda je považována za spolehlivou a také velice efektivní metodu přímého hledání[3].

2 Důkaz v [4, s.326] s odkazem na M.J.D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, Computer Journal, Vol. 7, No. 4, pp. 303–307,1964.

3 V [3] jsou uvedeny odkazy na evidenci prokazující toto tvrzení.



Obrázek 2.8: Powellova metoda. Znázornění iteračního procesu a determinace následně použitého směru.

Simplex metoda

Metoda vychází z pozorování, že nejmenší počet designů pro vyhodnocení **N** dimenzionální úlohy je simplex. Simplex je geometrický útvar – mnohostěn – složený z **N**+1 bodů, které jsou jeho vrcholy. Ve 2-dimenzionální úloze se tedy jedná o trojúhelník, ve 3-dimenzionální je to čtyřstěn. Simplex s konstantní vzdáleností mezi vrcholy je pravidelný simplex.

Základní myšlenka, navržená *Spendleym*, *Hextem* a *Himsworthem* je ta, že po vyhodnocení všech vrcholů pravidelného simplexu se vyřadí ten s nejhorší funkční hodnotou a nahradí se novým vrcholem reflektovaným na opačné straně. Postup byl vylepšen *Nelderem* a *Meadem*. Ti se neomezili na podmínku pravidelnosti simplexu a umožnili tak aby se nový bod nacházel na místě s předpokladem co možná nejnižší funkční hodnoty. Takto umožněná změna tvaru geometrie znamená, že simplex může být zvětšen (*expanze*) nebo zmenšen (*kontrakce*).

Na výpočet vrcholů rovnoměrného simplexu se používají následující vzorce [9]:

$$x_j = x_0 + p \cdot e_j + \Sigma\, q \cdot e_k, \quad j = 1, \ldots, n \tag{2.26}$$

Pro sumu od **k = 1** do **n** při podmínce **k ≠ j**.

$$p = \frac{a}{n\sqrt{2}}\left(\sqrt{n+1} + n - 1\right) \tag{2.27}$$

$$q = \frac{a}{n\sqrt{2}}\left(\sqrt{n+1}-1\right) \tag{2.28}$$

kde $\mathbf{e_k}$ je jednotkový vektor podél **k**-tého souřadnicového směru a $\mathbf{x_0}$ je počáteční vztažný bod.

Operace *reflexe* vytváří nový bod **x** podél čáry procházející z $\mathbf{x_h}$ do těžiště **x'** zbylých bodů, kde těžiště je

$$x' = \frac{1}{n}\Sigma\, x_i \; pro\; i \neq h \tag{2.29}$$

Vrcholový bod na konci *reflexe* bude

$$x_r = x' + \alpha\cdot\left(x' - x_h\right) \tag{2.30}$$

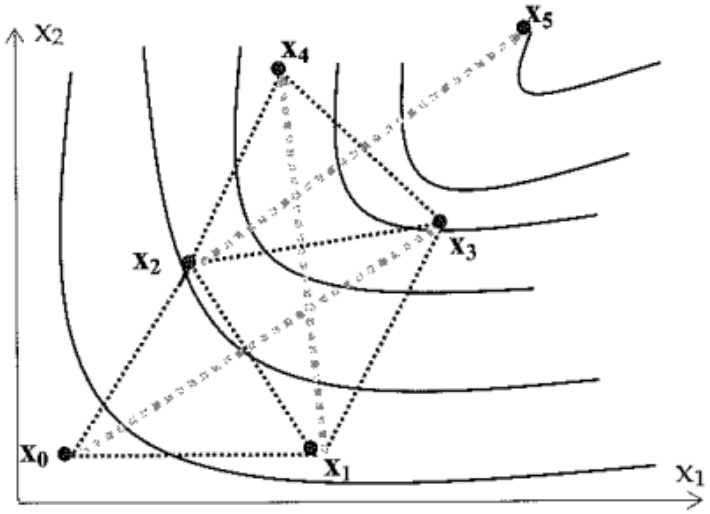
podobně *expanze*

$$x_e = x' + \beta\cdot\left(x' - x_h\right) \tag{2.31}$$

a konečně *kontrakce*

$$x_c = x' + \gamma\cdot\left(x' - x_h\right) \tag{2.32}$$

Koeficienty ve vzorcích (2.30),(2.31) a (2.32) jsou koeficienty této metody. *Nelder* a *Mead* doporučili hodnoty $\alpha = 1$, $\beta = 2$, $\gamma = 0.5$ [3].



Obrázek 2.9: Simplex metoda v 2D optimalizaci

Simplex metoda patří mezi heuristické metody a není možno matematicky prokázat její efektivitu. V praktických příkladech se však tyto metody osvědčují. *Simplex*

metoda je vhodná pro úlohy s počtem proměnných menším jak 10, jinde se uvádí, že konverguje pomalu při počtu proměnných větším jak 40 [17].

2.4.2. Metody vyžadující hodnotu 1.derivace

Gradient funkce

Gradient je směr definující nejrychlejší stoupání funkce (a opačný směr je tedy nejrychlejší klesání). Pokud se v n-dimenzionální úloze pohybujeme ve směru gradientu, hodnota funkce roste nejvyšší možnou rychlostí. Jedná se však pouze o lokální vlastnost, která se mění se souřadnicemi a má definici

$$\nabla f = \begin{Bmatrix} \delta f / \delta x_1 \\ \delta f / \delta x_1 \\ .. \\ .. \\ \delta f / \delta x_n \end{Bmatrix} \tag{2.33}$$

Cauchyho metoda největšího spádu („*Steepest descent method"*)

Cauchy byl první kdo použil záporný směr gradientu pro minimalizaci funkce. Princip metody je hledání optima ve směru záporného gradientu s jeho opravou v každé iteraci.

Postup:

1. Vyhodnocení ve startovním bodě $\mathbf{X_1}$ pro iteraci **i = 1**

2. Nalezení směru hledání $\mathbf{S_i}$ ve tvaru

$$S_i = -\nabla f_i = -\nabla f_i(X_i) \tag{2.34}$$

3. Vypočíst optimální délku kroku $\boldsymbol{\lambda_i^*}$ pro směr $\mathbf{S_i}$ a určit

$$X_{i+1} = X_i + \lambda_i^* S_i = X_i - \lambda_i^* \nabla f_i \tag{2.35}$$

4. Otestovat nový bod na optimálnost a pokud je dostatečný ukončit iterační proces

5. Nová iterace, návrat ke kroku č.2

Konvergenční kritéria, která mohou být použita pro ukončení optimalizace:

1. Dostatečně malá změna funkční hodnoty mezi dvěma po sobě jdoucími iteracemi
2. Dostatečně malá hodnota některé komponenty gradientu
3. Příliš malá změna návrhové proměnné mezi iteracemi

Metoda sdružených gradientů (Fletcher-Reevesova metoda)

Konvergenci *Cauchyho metody* lze dále vylepšit úpravou na *metodu sdružených gradientů*. Tím se vylepší konvergence na kvadratickou jak již bylo zmíněno u *Powellovy metody*. Předpokládá se kvadratický tvar funkce

$$F = \frac{1}{2} x^T H + b^T x + c \tag{2.36}$$

Substitucí (2.35) do (2.36) získáme potřebnou délku kroku

$$\lambda^* = \frac{-\left(x_k^T H + b^T\right) s}{s^T H s} \tag{2.37}$$

Kde **H** je *Hessova čtvercová matice*.

Stručně řečeno *Fletcher-Reevsova metoda* využívá sdružených směrů v kombinaci s gradientem.

Metoda je sice nadřazená *metodě největšího spádu* či *metodě předlohového hledání*, ale zároveň je méně efektivní než *Newtonova* či *kvazi-Newtonova metoda* [4,s.343].

Kvazi-Newtonova metoda

Iterační proces je dán iteračním procesem *Newtonovy metody* (2.39)

Princip metody spočívá v tom aproximovat (invertovanou) *Hessiánovou* matici bez použití druhých derivací nahrazením jiné matice **[B]** nazývané *metrická matice*. Způsobů výpočtu matice **[B]** v **i**-tém kroku je více a lze je dohledat například v [4, s.351].

Metoda je velice efektivní a její numerickou interpretací je metoda pojmenovaná podle jejích objevitelů *Broyden–Fletcher–Golgfarb–Shanno* (zkráceně *BFGS*). Více o ní a její nízko-paměťové variantě *L-BFGS* bude řečeno v kapitole 3.2.1.

2.4.3. Metody vyžadující hodnotu 2.derivace

Newtonova metoda

Jedná se vlastně o rozšíření jedno-dimenzionální formy *Newtonovy metody*. Rovnici (2.17) lze psát v maticovém tvaru jako

$$f(X)=f(X_i)+\nabla f_i^T(X-X_i)+\frac{1}{2}(X-X_i)^T[H_i](X-X_i) \tag{2.38}$$

Kde $\mathbf{[H_i]}$ je matice parciálních derivací 2.řádu vypočtená v bodě $\mathbf{X_i}$.

Zjednodušením této rovnice získáme aproximované řešení

$$X_{i+1}=X_i-[H_i]^{-1}\nabla f_i \tag{2.39}$$

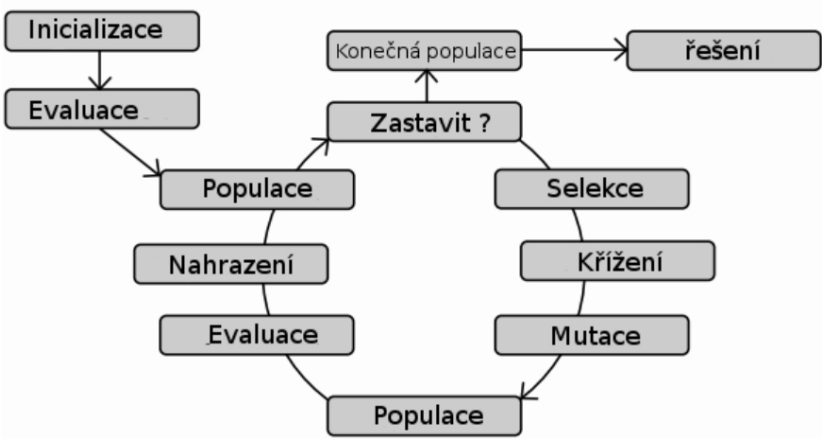
Vzhledem k výpočetní náročnosti MKP výpočtů je však získávání druhých derivací výpočetně neúnosné, proto se metoda nepoužívá. Používá se však její aproximace *kvazi-Newtonova metoda* zmíněná výše.

2.4.4. Stochastické metody

Jsou metody založené na náhodném prohledávání.

Evoluční algoritmy

Tento pojem je všeobecně používaný k označení meta-heuristického populačního optimalizačního algoritmu, který využívá postupy inspirované biologickou evolucí, jako jsou *reprodukce*, *mutace, překombinování*, *dědičnost*, *přirozený výběr* a *přežití nejsilnějších jedinců*. *Evoluční algoritmy* pracují s náhodnými změnami navrhovaných řešení. Nepoužívají derivace optimalizované funkce, pouze funkční hodnoty. Kandidátské řešení optimalizačního problému hraje úlohu jednotlivců v populaci a účelová funkce simuluje prostředí, ve kterém dané řešení „žije“. Evoluce populace se pak děje uskutečňováním opakované aplikace uvedených genetických operátorů. Pokud jsou v iteračním procesu nová řešení výhodnější, nahrazují se [9].



Obrázek 2.10: Typický průběh evolučního algoritmu [8]

Typicky používaným zástupcem je skupina *genetických algoritmů*.

Charakteristiky genetických algoritmů

Vhodné pro globální optimalizaci, pro výpočty diskontinuálních a diskrétních proměnných.

Vyšší počet vyhodnocení funkce k nalezení minima.

Roj částic

Je založena na vzoru chování kolonie, či roje hmyzu jako jsou mravenci, včely, hejno ryb či ptáků apod. Metoda simuluje chování těchto sociálních organismů. Každé individuum (v optimalizaci nazýváno *částice*) skupiny se chová podle vlastní inteligence a zároveň podle inteligence celé skupiny. Pokud tedy jedna *částice* nalezne cestu k jídlu, ostatní ji budou okamžitě následovat i přestože jsou v roji daleko. [4]

Z pohledu *více-dimenzionální optimalizace* je za roj považována skupina *částic* o konečném počtu s náhodně určenou počáteční polohou. Každá *částice* je popsána svojí polohou a rychlostí. Každá se pohybuje v návrhovém prostoru a pamatuje si nejlepší vhodnou pozici (z hlediska hodnoty objektivní funkce). Částice si tyto informace vyměňují a upravují podle toho svoji polohu a rychlost (v další generaci).

Stejně jako v přírodě optimalizační roj napodobuje určité chování, které je založeno na následujících faktorech:

1. *Soudržnost* – roj drží pospolu
2. *Separace* – nepřibližují se navzájem příliš k sobě
3. *Uspořádání* – následují obecný směr roje

Vhodná oblast aplikace této metody je podobná jako u genetických algoritmů: nelineární a diskrétní úlohy.

2.5. Shrnutí a volba metod pro řešení testovacích úloh

U statických lineárních úloh lze předpokládat, že častým jevem bude hladký průběh objektivní funkce a jedno globální minimum. Pro takový druh úloh je vhodná kvazi-Newtonova metoda z metod 2.řádu, Cauchyho metoda z metod 1.řádu a Simplex metoda (0.řádu).

Pokud se očekává nelineární chování, měly by využití spíše *genetické algoritmy* a *roj částic*, které mají v nelineárních úlohách tu výhodu, že nevyřešení libovolného kroku neznamená ukončení optimalizace tak jako v gradientních metodách a navíc jsou schopny lépe prohledat prostor, ačkoliv na úkor rychlosti konvergence.

3. Rešerše optimalizačních programů

V této kapitole bude krátce shrnuto jaké programy jsou dostupné pro řešení tvarové optimalizace.

3.1. Průzkum trhu

Při hledání softwarových balíků vhodných pro řešení strukturálních úloh s MKP bylo zjištěno, že na trhu se vyskytují programy, které se zabývají různými oblastmi optimalizace konstrukčního návrhu:

- *Topologická optimalizace* – s rostoucí kapacitou výpočetních prostředků a pokročilých výrobních technologií se stává velice žádanou.
- *Průzkum designu* - průzkum prostoru návrhu spolu s citlivostní analýzou (analýza zjišťující vliv jednotlivých parametrů), všeobecně použitelný základní nástroj optimalizace návrhu.
- *Tvarová optimalizace* – řešení optimálního tvaru návrhu způsobem hledání nejvhodnější kombinace hodnot parametrů parametrizovaného modelu.
- *Analýza spolehlivosti a bezpečnosti* – stochastický přístup, který zhodnocuje náhodné vlivy a jejich pravděpodobnostní rozložení. (používaný též výraz „*Six Sigma*“)

Bylo nalezeno několik programů zabývající se různými oblastmi problematiky optimalizace:

TOSCA Structure [12]

Zabývá se *topologickou optimalizací* ve strukturální analýze. Proces návrhu začíná návrhem optimalizovaného prostoru a pokračuje přes topologií a vyhlazení křivek a napěťových vrcholů až po vyrobitelnost. Program nemá vlastní řešič a využívá propojení se běžně užívanými MKP programy.

Genesis [13]

Zaměřený na optimalizaci návrhů v oblasti strukturální analýzy (statická, modální, přenos tepla, buckling). Program má vlastní řešič a firma dodává zvlášť i balík pre- a post- processingu. Balík se zabývá tvarovou i topologickou optimalizací a při návrhu lze zohlednit i výrobní možnosti (např. lití). Na webu výrobce -společnosti VR&D- je přímo uvedena nabídka krátkodobého zapůjčení programu.

Isight [14]

Program zaměřený na procesní simulaci, dynamické chování soustav, průzkum designu a obsahuje i matematický nástroj pro parametrickou optimalizaci *MISQP*, což je varianta skupiny metod sekvenčního kvadratického programování (*SQP*).

SmartDO [15]

Nástroj pro matematickou optimalizaci, který neobsahuje vlastní MKP řešič. Program se soustředí na řešení reálných úloh tvarové a topologické optimalizace. Z tvarové optimalizace jsou zde k dispozici tradiční nástroje NLP („non-linear programming“), *genetické algoritmy* a *roj částic*. Propaguje také vlastní přístup heuristické optimalizace („*Smart Heuristic Search Technology*“). Jedná se o Tchajvanskou firmu.

Výběr programů na testování

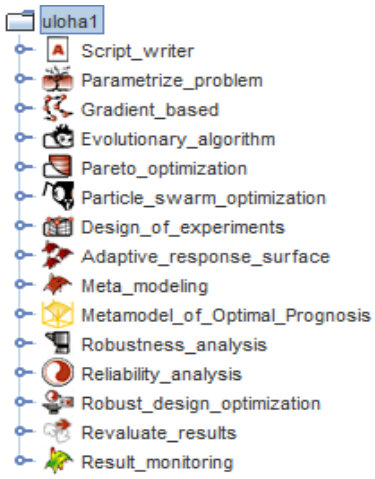
Za účelem zapůjčení optimalizačního softwaru bylo autorem osloveno několik firem. Jednalo se o *FE-Design* a distribuční firmu *TechSoft Engineering s.r.o.* (pro program *TOSCA Structure*), distribuční firmu *SVS FEM s.r.o.* (*optiSLang*) a *Vanderplaats Research & Development, Inc.* (*Genesis*). K dispozici byly na konci jednání programy *Genesis* a *optiSLang*. Z důvodu časové omezenosti pro tvorbu této práce byl jako reprezentant výkonného komerčního programu vybrán program *optiSLang.*

3.2. OptiSLang [16]

OptiSLang je čistě matematický nástroj (tj. bez vlastního MKP řešiče) používaný pro úlohy týkající se citlivostní analýzy, multidisciplinární optimalizace, analýz spolehlivosti a odolnosti, a také optimalizace robustnosti systému. Jedná se o produkt firmy *Dynardo GmbH*. V této podkapitole budou probrány optimalizační možnosti programu a uživatelské prostředí.

Popis prostředí

Hlavní strom projektu se skládá z části pro přípravu modelu, metod, které již byly po teoretické stránce popsány v kapitole 2.4 (gradient based, Evolutionary alg., Particle swarm) a dalších, doposud v práci nezmíněných, metod. Zaměřme se nyní na poslední skupinu aby bylo možné posoudit jaké další nástroje jsou uživateli k dispozici.



Obrázek 3.1: optiSLang - prostředí - nabídka

3.2.1. Dostupné metody [8]

Gradient based

Gradientní metody jsou zde dvě: *NLPQLP* a *L-BFGS*. Obě jsou metody 1.řádu a jsou založeny na Newtonovském přístupu (viz kap.2.4.2).

L-BFGS je modifikovaná *kvazi-Newtonova metoda BFGS*. „L“ značí *Limited Memory.* Metoda je využívána na řešení rozsáhlých nelineárních optimalizačních úloh.

NLPQLP je označení pro „Nonlinear programming using a Quadratic or Linear Least-square“. Jedná se tedy o metodu patřící do skupiny SQP metod, které hledají řešení způsobem kvadratického programování („*Quadratic programming*“) podobné

rovnici 2.36. Je vhodná pro nelineární úlohy s omezeními s hladkým průběhem funkce. Od základního *SQP* algoritmu se liší schopností překonat určité chybové situace, kvůli kterým by jinak iterační proces selhal.

Evolutionary algorithm

Evoluční algoritmy (EA) začínají náhodnou populací *n* potomků a každá další generace vzniká na základě této předchozí generace za pomocí evolučních postupů zmíněných v kapitole 2.4.4. *OptiSLang* nabízí nastavení genetického algoritmu na dva režimy: globální a lokální (v [9] pojmenován „*design improvement*“).

Využití má především tam, kde selhávají gradientní metody a metoda odezvy povrchu („*response surface“)*. *EA* je velmi robustní a odolný vůči špatně podmíněným úlohám. Využívá se také u úloh s diskrétními veličinami a u úloh s velkým počtem proměnných. V optimalizaci není nic zadarmo a *EA* si vybírá svou daň pomalou konvergencí.

Pareto optimization

Pareto je způsob (původem převzatý z ekonomie) jakým se vypořádat s multi-objektivní optimalizací. Jedná se spíše o způsob jak efektivně vyhodnotit, který z navržených designů je lepší pokud má úloha více vyhodnocovacích kritérií, které jsou navzájem v konfliktu. Aplikace *Pareto* v optiSLangu je využívána v oblasti *EA*.

Particle swarm optimization

Roj částic, popsaný v kapitole 2.4.5, je zde využíván, podobně jako *EA*, globálním a lokálním přístupem. Globální hledání způsobuje, že částice hledají spíše ve směru jejich vlastního nejlepšího řešení, kdežto lokální hledání zvažuje nejlepší řešení celé skupiny (*leadera* skupiny), a nejlepší řešení oné částice, rovnoměrně. Výběr přístupu ovlivňuje nastavení inicializace, adaptace a ukončovacího kritéria hledání.

V testovacích úlohách bude použito nastavení globálního hledání.

***Design of experiments** (DOE)*

Odezva a korelace proměnných. Tvorba podpůrných bodů (např. pro *meta modeling)*.

***Adaptive response surface** (ARS)*

Je interpolační přístup k vystižení tvaru funkce polynomem v n-dimenzionálním prostoru. Přístup je vhodný spíše pro hladké funkce s jedním minimem v optimalizovaném prostoru.

Meta modeling

Vytvoření nahrazujícího modelu z podpůrných bodů, které mohou být importovány z jakékoliv jiné analýzy (ale obvykle z DOE) nebo z reálných experimentů. Tento nástroj nevyhodnocuje funkci (a tedy nekomunikuje s externím výpočetním programem), ale je to čistě matematický nástroj sloužící k popsání chování skutečného modelu. Jedná se vlastně o volnější podobu metody *Adaptive Response Surface,* ale je zde prostor pro manuální nastavení podpůrných bodů a popisu nahrazené funkce modelu.

Meta model of optimal prognosis

Nástroj slouží k posouzení spolehlivosti náhradního modelu spočteného pomocí *Meta Modeling*. Část vstupních experimentů použije na aproximaci modelu, část použije na ověření korelace aproximovaného meta modelu s daty, které nebyly k jeho aproximaci použity. Výsledkem je koeficient („*coefficient of prognosis*“), který kvantifikuje onu korelaci.

Robustness analysis

Analýza náchylnosti návrhu k selhání vlivem stochastických odchylek návrhových proměnných (a jiných vlivů).

Reliability analysis

Slouží pro zjištění pravděpodobnosti selhání.

Robust design optimizations

Uvažuje stochastické jevy (parametrů) a vliv těchto jevů na funkcionalitu návrhu. Tato metoda se snaží optimalizovat návrh tak, aby nebyl citlivý na náhodné jevy. Výsledkem je tedy návrh s minimální pravděpodobností selhání. Identifikuje také parametry, které nejvíce ohrožují funkcionalitu návrhu.

Shrnutí funkcionality nástrojů programu optiSLang

Nástroje můžeme shrnout do dvou skupin

1. přímá optimalizace: gradient based, EA, particle swarm, ARS
2. podpůrné optimalizační nástroje: pareto optimization, DOE, meta modeling, meta model of optimal prognosis

Volba testovaných metod

Pro účel porovnání optimalizace s algoritmy v *Matlabu* budou z dostupných optimalizačních prostředků v programu *optiSLang* využity a testovány gradientní metody *NLPQLP* a *L-BFGS* pro porovnání se stejnými algoritmy dostupnými v programu *Matlab*. Dále byly vybrány známé stochastické přístupy: *genetický algoritmus* a *roj částic* pro porovnání jejich efektivity na inženýrských úlohách s *gradientními metodami*.

Další programy, které jsou přímo použity při testování, jsou popsány dále.

3.3. Matlab

Matlab, jakožto výkonný matematický nástroj, obsahuje ve svém „*Optimization*

Tool" širokou paletu optimalizačních algoritmů. Z nich bude v analýze testovacích úloh využit pouze *BFGS* (jako zástupce metod 2.řádu) a *L-BFGS* (paměťově méně náročná verze *BFGS*), který poslouží pro přirovnání s *L-BFGS* algoritmem implementovaným v *optiSLangu*.

3.4. ANSYS

ANSYS obsahuje několik málo optimalizačních metod. Jedná se především o „*Subproblem aproximation method"* (metoda 0.řádu) a o jednu gradientní metodu. Druhá jmenovaná metoda bude použita jen pro letmé porovnání testovací úlohy č.3 s efektivitou ostatních algoritmů.

ANSYS Workbench disponuje dalšími podpůrnými nástroji optimalizace

Parameters correlation **–** zjišťování korelace jednotlivých parametrů, což dobře poslouží k eliminaci parametrů, které nemají nezávislý vliv.

Response surface **–** odezva objektivní funkce na jednotlivé parametry (nesouvisí s „*Adaptive Response Surface").*

Six Sigma Analysis **–** pravděpodobnostní přístup k ověření robustnosti návrhu v reálných podmínkách (materiálové vlastnosti, geometrie, technologické úpravy – nikdy nejsou konstantní).

Shape optimization **–** uživatelsky nenáročná topologická optimalizace.

Propojení optimalizačního softwaru a programu ANSYS lze provést pro obojí APDL, tak i Workbench prostředí [5],[6]. Pro APDL je to otázka zapsání v Matlabu souboru s parametry, který je v dávkovém souboru načten.

V případě prostředí *Workbench* je situace o něco složitější. *Workbench* je nutné spustit s užitím předpřipraveného kódu psaného jazykem Python, který Workbench vnitřně používá na komunikování příkazů (a je v něm zapisován i „*journal*"). Připravil jsem si skript, který načte proměnné ze souboru, obnoví projekt a vypíše požadované výsledky do výstupního souboru. To poskytuje výhodu v náročnějších úlohách, kde je *APDL* zápis nežádoucí. *Workbench* má však nevýhodu toho, že nad rámec výpočetního

času mu zabere téměř minutu obnovení projektu po změně proměnných. Soubory nebyly nakonec použity, přikládám je alespoň na diskovém médiu pro další zájemce.

3.5. Shrnutí a volba programů pro řešení testovacích úloh

Tato práce je zaměřena na porovnání výkonnosti algoritmů uživatelem připravených a těch dostupných v komerčním programu. Z programu *optiSLang* je metodám dostupným v *Matlabu* (a těm jednodušším, které si uživatel může napsat relativně rychle sám) podobný algoritmus *L-BFGS* a *NLPQLP* (jakožto obdoba *BFGS*). Dále byly pro přídavné porovnání s výsledky dosaženými *gradientními metodami* a *metodou Simplex* vybrány dva tradičnější stochastické nástroje *evoluční algoritmus* a *roj částic*.

Program *optiSLang* i *Matlab* obsahují mnoho dalších optimalizačních algoritmů avšak jejich porovnávání, nebo hledání nejlepších dostupných metod není cílem této práce.

4. Testování metod a optimalizačních programů

Bylo vybráno několik lineárních statických strukturálních úloh, aby bylo možné určit, který způsob je pro řešení definované oblasti úloh efektivní: Jednoduchá teoretická úloha s exaktním řešením, zjednodušená reálná a reálná úloha z praxe.

Byly vybrány úlohy jednoduššího typu s nízkým počtem optimalizovaných parametrů a malou výpočetní náročností. To umožňuje provádět více experimentálních výpočtů čímž lze vylepšit míru spolehlivosti testování a posuzování výsledků. Například pro úlohy o třech optimalizovaných parametrech lze vizualizovat *průzkum prostoru* řešení a udělat si tak představu o průběhu funkce v celém prostoru proměnných.

4.1. Metodika testování

Výběr metod

Na konci kapitol 2 a 3 byly vybrány algoritmy jež budou testovány. Tyto jsou:

1. *Matlab* – Simplex (Nelder-Mead)
2. *Cauchy – Zlatý řez*: Cauchyho gradientní metoda v kombinaci s m.Zlatého řezu
3. *Cauchy – interpolace*: Cauchyho grad. metoda v kombinaci s 1D interpolací
4. *Matlab – BFGS*: gradientní metoda implementovaná v Matlabu pomocí funkce *fmincon*
5. *Matlab – L-BFGS*: gradientní metoda implementovaná v Matlabu pomocí vestavěné funkce *fmincon*
6. *optiSLang* – gradientní metoda LBGFS
7. *optiSLang* – gradientní metoda NLPQLP
8. *optiSLang* – evoluční algoritmus

9. *optiSLang* – roj částic (globální prohledávání)

4.1.1. Tvorba algoritmů – časová náročnost

Autorem byly vytvořeny či upraveny algoritmy pro *metody Simplex*, *Cauchy* a *metodu Zlatého řezu* a *kvadratickou interpolaci*, které byly nezbytné pro část jednodimenzionálního hledání *Cauchyho metody.* Úpravy v kódu *Cauchyho* a *Simplex metody* spočívaly pouze v ošetření pro případ omezeného hledání (byla implementována jednoduchá funkce *checklimits.m*, která kontrolovala, a v případě potřeby opravovala, hodnoty na limitní hranici intervalu před vyhodnocením funkce v uvedených algoritmech). I toto pouhé ošetření si vyžádalo několik dní práce a nutnost absolutního porozumění struktuře algoritmů.

Z toho lze usoudit, že napsání algoritmu pro jedinou metodu je několika denní práce a v případě implementace přídavných ošetřujících opatření pro větší robustnost a ladění kódu by se tvorba kvalitního algoritmu mohla protáhnout i na několik týdnů.

4.1.2. Nastavení algoritmů

Podstatnou roli ve výkonu metody mohou hrát volitelné parametry, které určují její chování. Na nejvhodnější volbu parametrů algoritmů se autor práce nezaměřoval, avšak je si vědom, že to může znatelně ovlivnit průběh celého testování. Vzhledem k tomu, že takové vyladění je časově náročné, předpokládejme férovou podmínku, že nebylo zasahováno do nastavení žádného z algoritmů tvořených v programu Matlab ani do nastavení algoritmů v programu *optiSLang* (ponecháno původní). Pokud už bylo něco pozměněno, bylo tak učiněno z nezbytnosti (např. délka kroku pro *BFGS* a *L-BFGS* funkce *fmincon* v *Matlabu,* jinak vůbec nefungoval). Kompletní nastavení lze vyčíst ze zdrojových kódů přiložených na datovém médiu, stejně tak lze vyčíst ze souborů nastavení programu *optiSLang*.

4.1.3. Metodologie vyhodnocení

Vyhodnocuje se především schopnost algoritmu dosáhnout nejlepšího řešení spolu s počtem iterací, za který bylo toto řešení dosaženo. Požadovaná vyšší přesnost výsledku (kritérium ukončení optimalizace některých algoritmů) sebou nese zvýšený počet

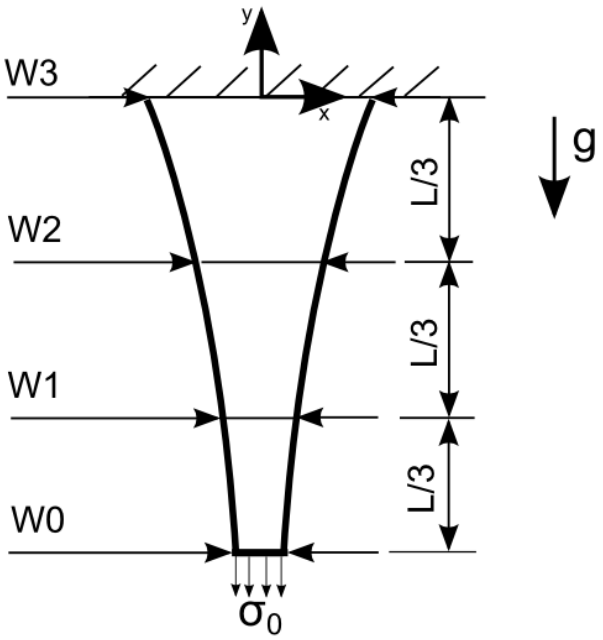
iterací. Nebylo však vždy možné nastavit kritérium zastavení výpočtu na určité hodnotě objektivní funkce. Mimo jiné i z toho důvodu, že některé metody by takového výsledku nemusely dosáhnout a jejich výsledek by byl znehodnocen. Proto je při posuzování výkonnosti brána v úvahu i celková konvergence metod, která je provedena na základě průběhu objektivní funkce v iteračním procesu. Všechny tyto průběhy jednotlivých algoritmů jsou uvedeny v příloze.

4.2. Popis testovacích úloh a výsledky testování

Pro porovnání byly zvoleny následující tři úlohy:

1. Jednoduchá testovací úloha nosníku konstantní pevnosti (dále jen „testovací úloha 1“ nebo TU1) - slouží především pro ověření funkčnosti algoritmu a jeho schopnosti přiblížit se přesnému řešení, které je pro tuto úlohu známo.
2. Velikost a poloha odlehčovacího vrubu na náboji nalisovaném na hřídeli - jedná se o úlohu popisující skutečnou problematiku omezení koncentrátoru napětí, který vzniká na přechodové hraně mezi nalisovaným nábojem a hřídelem. Úloha je poměrně náročným testem optimalizačních algoritmů, protože průběh funkce není ve zkoumané oblasti unimodální a algoritmus může snadno zůstat v lokálním minimu nebo divergovat.
3. Omezení vrubového účinku na hřídeli. Jedná se o skutečnou úlohu z praxe, která je z hlediska průběhu funkční hodnoty podobná TU1 s malou měrou globální-lokální problematiky.

4.2.1. Testovací úloha 1 – nosník konstantní pevnosti



Obrázek 4.1: Testovací úloha č.1 - nákres

Popis úlohy: Nosník je pevně uchycen na jednom konci v homogenním gravitačním poli tak aby visel. Na volném konci je aplikováno na průřezu tahové napětí. Obě složky zatížení tedy vyvozují ve směru podélné osy nosníku pouze tahové napětí. Cílem je nalezení takového tvaru nosníku pro který bude osové napětí v celém nosníku konstantní. Pro tuto úlohu je známé analytické řešení.

Geometrie: Nosník je modelován jako rovinná úloha. Tvar geometrie určuje křivka typu *spline* se 4-mi řídícími body rovnoměrně rozloženými po délce nosníku. Šířky nosníku jsou označený jako W0 až W3.

Doba jednoho výpočtu je přibližně 5 sekund.

Optimalizace

Tvar nosníku je řízen třemi parametry – průřezem nosníku ve třech bodech W1, W2 a W3. Ostatní hodnoty jsou neměnné.

Z hlediska optimalizace je úloha specifická také tím, že definice tvaru pomocí *spline* křivky způsobuje konstrukční omezení – jistá kombinace parametrů totiž vytvoří plochu prolínající sebe sama, což má za následek selhání výpočtu.

Vyhodnocovací kritérium: Byl zvolen kvadratický přístup. Hodnota objektivní funkce je suma čtverců odchylek osového napětí od předdefinovaného napětí σ_0 působícího na volném konci nosníku. Vyhodnocuje se 12 rovnoměrně rozmístěných bodů (uzlů) po celé délce nosníku (v ose symetrie).

$$F_{obj} = \sum_{i=1}^{n} (\sigma_{yi} - \sigma_{konst})^2 \tag{2.40}$$

Materiál	***Geometrie***	***Okrajové podmínky***
E = 210 000 MPa μ = 0,3 ρ = 7,85 10-8 t mm3 hustota zvýšena o 1 řád pro lepší grafické znázornění proměnné šířky (nosník by byl příliš štíhlý).	W0 = 10 mm L = 1000 mm parametry (poč. hodnoty a intervaly): W1 = 10 mm <8, 50> W2 = 10 mm <8, 50> W3 = 10 mm <8, 50>	Uchycení konce nosníku: ve směru y po celé šířce ve směru x v jediném bodě Takto definované uchycení lépe odpovídá předpokladům exaktního řešení. σ_0 = 1 MPa

Tabulka 4.1: TU1 – vlastnosti úlohy

Nejlepší řešení

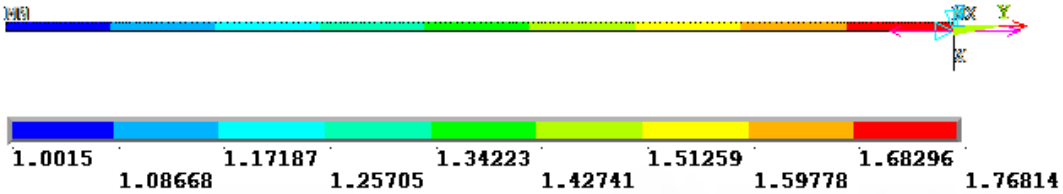
K této úloze je známo analytické řešení [10]

$$W(x) = W0 \exp\left[\frac{\gamma x}{\sigma_0}\right] \tag{2.41}$$

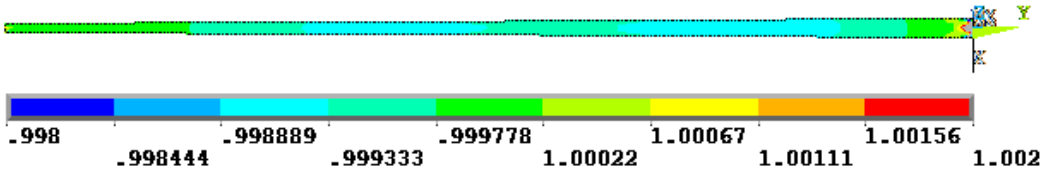
kde γ je měrná hmotnost a **x** proměnná délka nosníku.

Z analytického výpočtu vychází optimální řešení jako: W1 = 12.89, W2 = 16.62, W3 = 21.60. Při dosazení hodnot z analytického výpočtu získáme z MKP výpočtu hodnotu objektivní funkce Fobj = 2.89e-4. Jak je vidět v následující tabulce, nejnižší hodnoty objektivní funkce bylo dosaženo u gradientní metody LBFGS řešené v Matlabu.

Řešení



Obrázek 4.2: TU1 - počáteční stav nosníku s parametry [10 10 10] (vpravo uchycení, viz SS z obr.4.1), osové napětí [MPa]



Obrázek 4.3: TU1 - nejlepší řešení s hodnotami W1-W3 [13,01 16,76 21,60] (optiSLang - NLPQLP), osové napětí [MPa]

Výsledky

	Počet iterací	Fobj	Parametry		
			W1	W2	W3
Počáteční stav	1	2,07	10	10	10
Simplex Nelder-Mead	60	3,09e-4	12,84	16,58	21,59
Cauchy – zlatý řez	93	7,94e-3	12,84	17,20	19,25
Cauchy – interpolace	99	9,40e-4	12,91	16,76	20,72
Matlab - BFGS	96	8,58e-7	13,00	16,68	21,59
Matlab - LBFGS	**68**	**6,71e-7**	**13,00**	**16,76**	**21,58**
optiSLang - LBFGS	103	5,43e-7	13,00	16,75	21,59
optiSLang - NLPQLP	87	4,48e-7	13,01	16,76	21,60
optiSLang – evoluční	200	4,46e-3	13,38	16,47	21,86
optiSLang – roj částic	394	1,34e-5	12,99	16,78	21,55

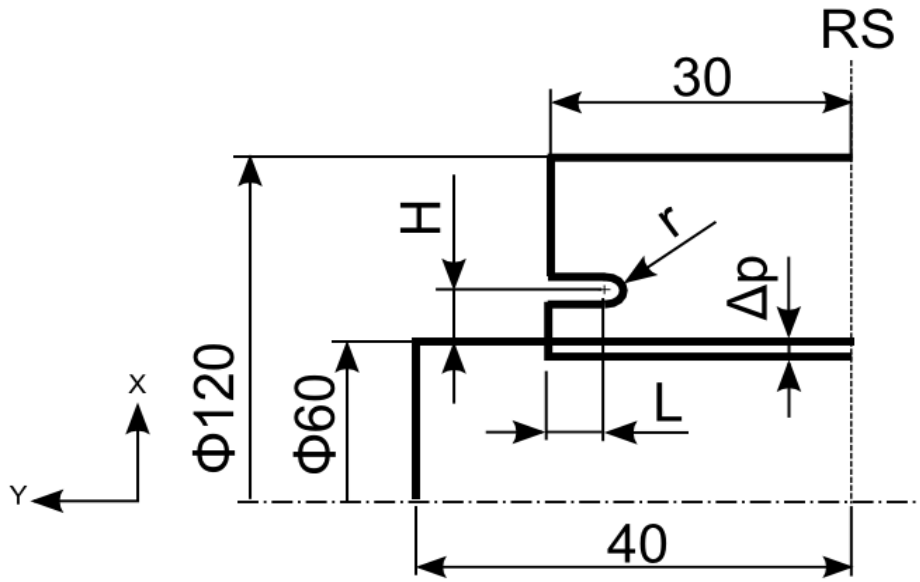
Tabulka 4.2: TU1 – souhrn výsledků testování

Shrnutí výpočtů TU1: Všechny metody konvergují ke správnému řešení. Všechny

gradientní metody mají podobnou výkonnost jak co do přesnosti řešení tak do počtu iterací s výjimkou *Cauchyho metody*, která konvergovala pomaleji a k méně přesnému výsledku, stejně jako *Simplex m.*, která má však o třetinu kratší výpočtový čas.

Nejlépe se jeví gradientní metoda *L-BFGS* v *Matlabu*, která dosáhla téměř nejpřesnějšího výsledku a za nejkratší čas. Naopak značně vysoký počet iterací s menší přesností výsledku vykazují *evoluční algoritmus* a *roj částic*.

4.2.2. Testovací úloha 2 – koncentrace napětí v nalisovaném spoji



Obrázek 4.4: Testovací úloha č.2 - nákres

Popis úlohy: Na hřídel je nalisován náboj (např. ozubené kolo) čímž vznikne koncentrace napětí na hřídeli pod okrajovou hranou náboje. Zvýšené napětí v této oblasti je nebezpečné z hlediska únavy. Cílem je tuto koncentraci napětí eliminovat a zajistit rovnoměrné rozložení napětí na celé stykové ploše. V této úloze je na náboji simulován zápich, který snížením tuhosti náboje v kritické místě sníží koncentraci napětí na hřídeli. Snížení tuhosti však snižuje velikost přeneseného kroutícího momentu a ten nesmí klesnout pod definovanou hodnotu $\mathbf{M_{kmin}}$.

Uchycení těles v prostoru je na ose rotace (ve směru **x**) a na ose symetrie **RS** (ve směru **y**) viz obr.4.4. Zatížení je vyvozeno přesahem nalisované hřídele. Je dána tolerance **H7** a **p6** což způsobuje minimální přesah 0,023 mm. Kontakt s přesahem byl

realizován v ANSYS/APDL pomocí *constraint equations*. Protilehlé uzly jsou svázány v radiálním směru následovně

$$UX_{hřídel} - UX_{náboj} = \Delta p \tag{2.42}$$

Geometrie: Úloha je modelována jako axisymetrická s další rovinnou symetrie kolmou na osu hřídele. Geometrie je popsána na obr.4.4. Optimalizované parametry definující geometrii zápichu jsou označeny ***H***, ***L*** a ***r.*** Ty definují vzdálenost zápichu od stykové plochy, hloubku a výška zápichu (definovanou poloměrem který drážku zakončuje).

Doba výpočtu jedné iterace 5 sekund.

Optimalizace

Úloha je specifická nelineárním průběhem objektivní funkce. Z průzkumu prostoru (viz kapitola 4.2.4) je patrné, že prostor má mnoho lokálních minim a optimalizace v tomto prostoru je globálního charakteru.

Počáteční interval nejistoty je stanoven s cílem dovolit algoritmům prozkoumat celý geometrický prostor, limity jsou tedy nastaveny pouze tak, aby zápich „neopustil“ prostor náboje. Počáteční bod byl stanoven na nízké hodnotě nedaleko minima.

Vyhodnocovací kritérium: Hledá se taková kombinace parametrů, aby bylo redukované napětí na povrchu hřídele po celé stykové ploše pokud možno konstantní. Tato část podmínky je zjednodušena na tvar $\mathbf{f_1 = (\sigma_{max} - \sigma_{min})}$. Je také nutno přenést definovaný kroutící moment $\mathbf{M_{kmin}}$. Pokud $\mathbf{Mk < M_{kmin}}$ tak je objektivní funkce penalizována $\mathbf{f_2 = 10*(M_{kmin} - Mk)}$. **Mk** je vyhodnocen jako $\mathbf{Mk = \sigma_r * L_s * f}$. Kde $\mathbf{\sigma_r}$ je průměrné radiální napětí na stykové ploše, $\boldsymbol{L_s}$ je délka stykové plochy a ***f*** je tření. Celá objektivní funkce je **Fobj = f1 + f2**. Úloha se nezabývá napětím v náboji.

Materiál	*Geometrie*	*Okrajové podmínky*
E = 210 000 MPa μ = 0,3 tření f = 0,2	Δp = 0,023 mm parametry (poč. hodnoty a intervaly): H = 1 mm <1, 20> L = 1 mm <0,1, 20> r = 1 mm <0,5, 5>	Uchycení v rovinách symetrie: osa hřídele: UX = 0 rovina symetrie RS: UY = 0 Mk_{min} = 360 N mm

Tabulka 4.3: TU2 – vlastnosti úlohy

Nejlepší řešení

Za přirovnávací nejlepší řešení bude považováno:

Fobj = 4,49e-1, parametry H = 1,2, L = 2,1, r = 0,5

Toto řešení bylo získáno průzkumem prostoru jenž je popsáno v kapitole 4.2.4.

Výsledky

	Počet iterací	Fobj	Parametry		
			H	L	r
Počáteční stav	1	4,79e-1	1	1	1
Simplex Nelder-Mead	**8**	**4,67e-1**	**1**	**1,69**	**0,75**
Cauchy – zlatý řez	nekonverguje				
Cauchy – interpolace	nekonverguje				
Matlab - BFGS	nekonverguje				
Matlab - LBFGS	nekonverguje				
optiSLang - LBFGS	nekonverguje				
optiSLang - NLPQLP	23	4,73e-1	1	1,19	1,05
optiSLang – evoluční	nekonverguje				
optiSLang – roj částic	nekonverguje				

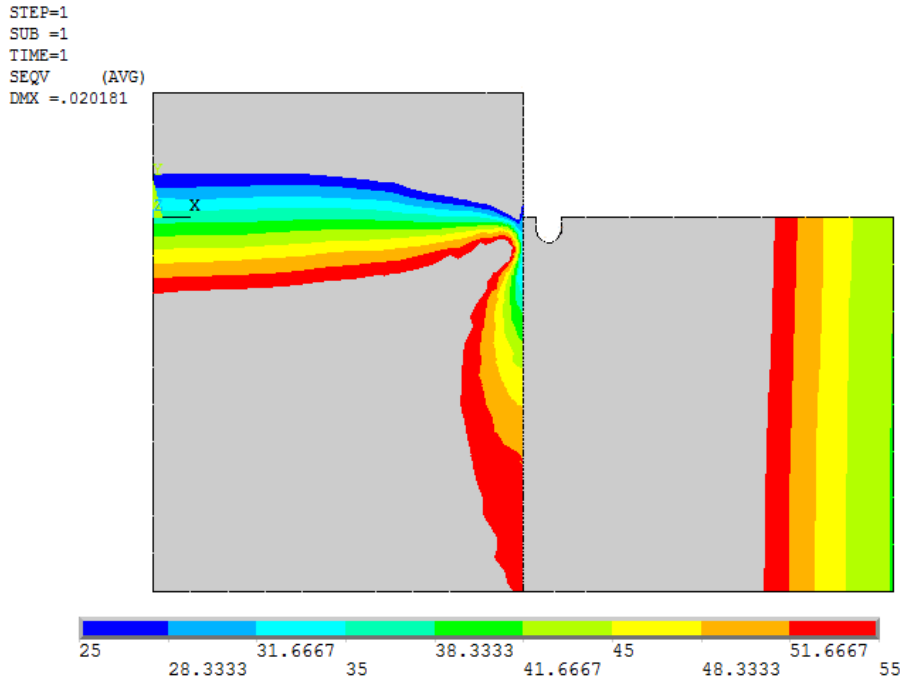
Tabulka 4.4: TU2 – souhrn výsledků testování

Shrnutí výpočtů TU2:

Testovací úloha a použitý startovní bod vedly k tomu, že většina metod divergovala. K lepšímu výsledku konvergoval za daných podmínek pouze *NLPQLP* a *Simplex metoda*.

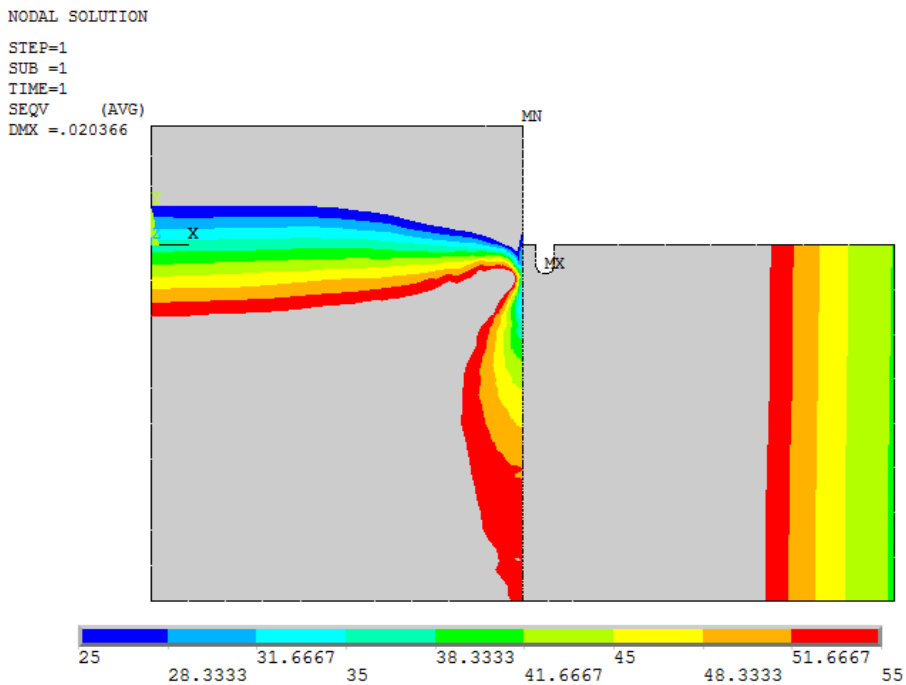
Aby byl vyloučen vliv náhody výběru počátečního bodu v prospěch jedné či druhé metody, byl proveden dodatečný test s jiným náhodně vybraným počátečním bodem [3 1 1] namísto [1 1 1]. Pro tento bod, který má **Fobj = 5,34e-1** algoritmus *NLPQLP* divergoval, zatímco *Simplex* konvergoval k hodnotě 4,83e-1 za 22 iterací (průběhy opět uvedeny v přílohách). V rámci provedených testů pro tuto úlohu je tedy Simplex nejstabilnější.

Pozn.: Byl také proveden test lokálního roje částic, který nalezl za 400 iterací řešení velice blízké nejlepšímu řešení. Tato varianta roje částic však v testování nebyla zahrnuta a tato informace je zde uvedena jen jako doplněk.



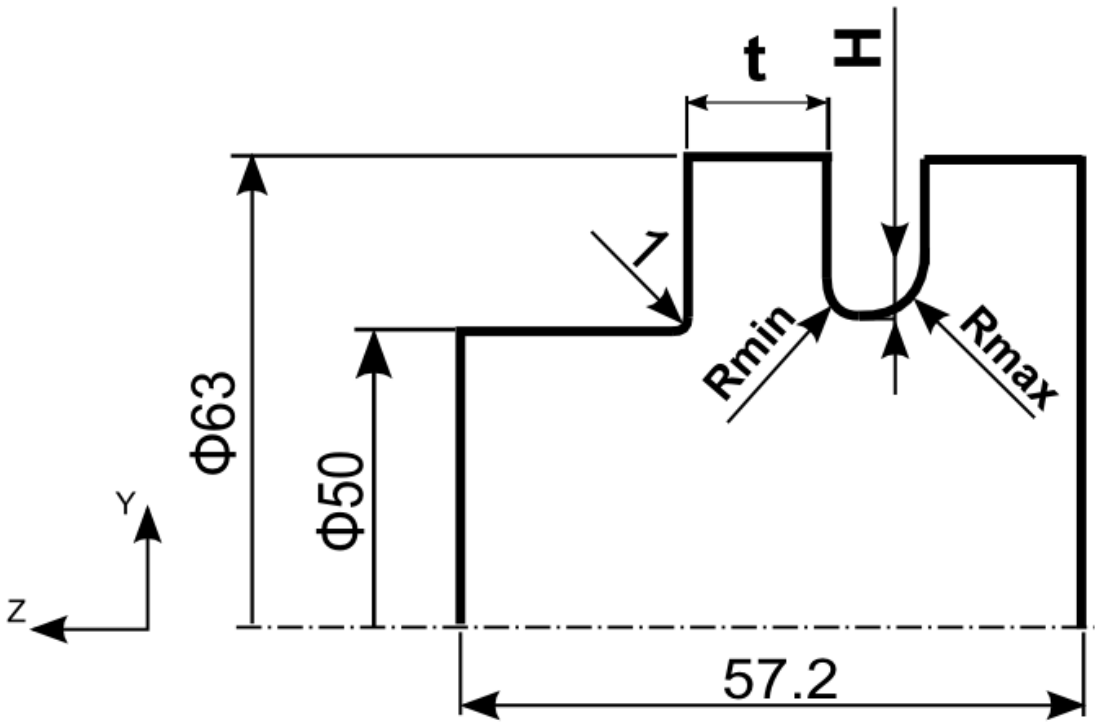
Obrázek 4.5: TU2 - počáteční stav [1 1 1], red.napětí HMH [MPa]

V přílohách je přiloženo řešení bez zápichu.



Obrázek 4.6: TU2 – nejlepší nalezené řešení [1 1,69 0,75], optiSLang – roj částic, red.napětí HMH [MPa]

4.2.3. Testovací úloha 3 – omezení vrubového účinku na hřídeli



Obrázek 4.7: Testovací úloha č.3 - nákres

Popis: Je dána hřídel o průměrech D1 a D2 a ostrým přechodem o poloměru **r**. U této lícní plochy je uvažována podmínka, že se zaoblení **r** nesmí změnit (např. proto, že kvůli ložiskovému domku je nutné dodržet rovnou opěrnou plochu). Cílem optimalizace je zmírnit koncentraci napětí v tomto přechodovém místě za pomocí zápichu na velkém průměru hřídele. Hřídel je zatěžovaná kroutícím momentem.

Tato úloha je skutečný technický problém řešený na zakázku[4]. Všechny parametry, rozměry a omezení odpovídají skutečnosti.

Geometrie: Úloha je 3-dimenzionální z důvodu aplikace kroutícího momentu. Rozměry viz úvodní nákres.

Doba výpočtu jedné iterace 50 sekund (*PCG* řešič).

Optimalizace

Technickým problémem je nalezení vhodné polohy a rozměrů zápichu. To je definováno 4-mi parametry: poloměry v zahloubení **Rmin** a **Rmax** (zleva), hloubka zápichu **hl** a vzdálenost zápichu **t** od čela velkého průměru hřídele.

Vyhodnocovací kritérium: Cílem je prosté snížení špiček napětí v okolí kritického místa. Minimalizuje se tedy maximální ekvivalentní napětí v modelu.

Materiál	***Geometrie***	***Okrajové podmínky***
E = 210 000 MPa μ = 0,3 tření f = 0,2	parametry (poč. hodnoty a intervaly): Rmin = 3 mm <1; 5> Rmax = 3 mm <1; 5> HL = 7 mm <5,5; 8> t = 3 mm <1,8; 5>	Mk = 600 Nm

Tabulka 4.5: TU2 – vlastnosti úlohy

4 Úlohu, včetně APDL kódu, poskytl vedoucí DP, Jan Szweda.

Nejlepší řešení

Nejlepší získané řešení je metodou *roj částic*:

Fobj = 64.75, parametry Rmin = 2.62, Rmax = 4.78, H = 6.59, r = 1.82

Toto řešení bylo získáno průzkumem prostoru jenž je popsáno v kapitole 4.2.4.

Výsledky

	Počet iterací	Fobj	Parametry			
			Rmin	Rmax	H	t
Počáteční stav	1	74,76	3,00	3,00	7,00	3,00
Simplex Nelder-Mead	**21**	**65,82**	**3,29**	**4,59**	**6,73**	**1,95**
Cauchy – zlatý řez	11	70,07	3,00	2,90	6,48	3,00
Cauchy – interpolace	14*	70,98	3,05	3,43	6,27	2,95
Matlab - BFGS	24	69,82	3,67	3,93	6,77	3,29
Matlab - LBFGS	27	69,34	3,36	3,51	6,71	3,10
optiSLang - LBFGS	121	66,06	4,50	4,46	6,82	1,80
optiSLang - NLPQLP	**35**	**66,14**	**5,00**	**5,00**	**6,98**	**1,80**
optiSLang – evoluční	193	65,04	2,65	4,40	6,57	1,80
optiSLang – roj částic	**137**	**64,75**	**2,62**	**4,78**	**6,59**	**1,82**
ANSYS	50	68,22	3,12	3,46	6,61	2,75

Tabulka 4.6: TU3 – souhrn výsledků testování

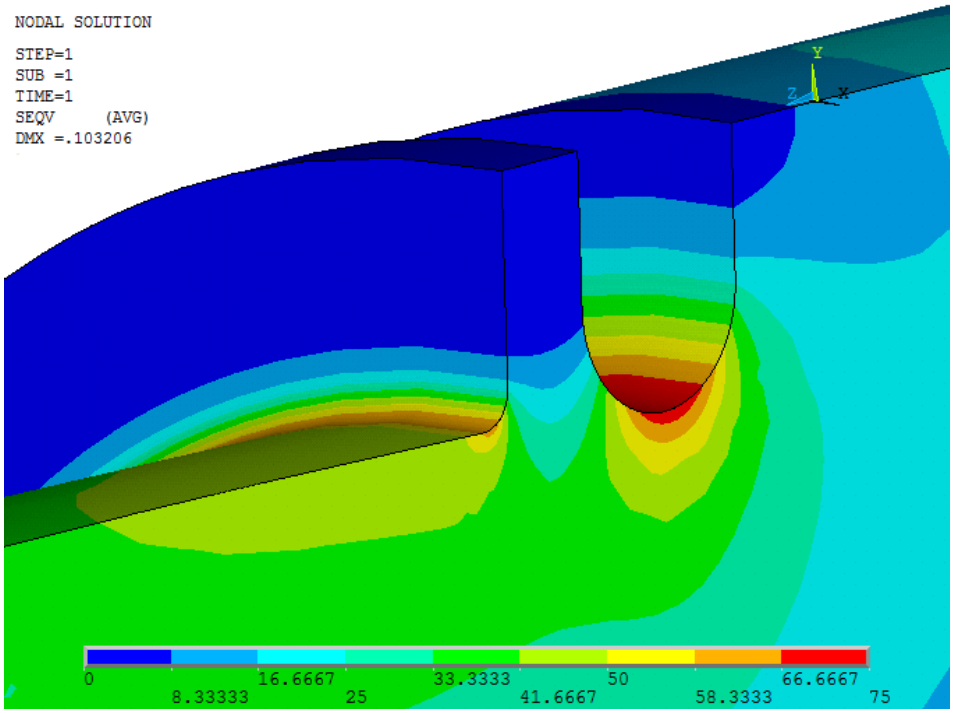
* výpočet zhavaroval v 18-té iteraci. Nejspíše z důvodu špatné definice makra nelze úlohu spočítat pro bod [3,1; 3; 7; 3]

Shrnutí výpočtů TU3: K nejlepšímu řešení se nejvíce přiblížily výpočty všech metod v *optiSLangu* a *Simplex metody* a to na hodnotu okolo 65 (což je hodnota obj. funkce přímo vyjadřující $\sigma_{red\text{-}max}$). Z výpočtů v programu *Matlab* měla nízký počet iterací a zároveň dobrý výsledek (Fobj = 65,82) *metoda Simplex*. Ostatní algoritmy konvergovaly pouze k řešení okolo hodnoty 70. To je pravděpodobně zapříčiněno lokálním minimem (více o průběhu funkce této úlohy v následující podkapitole), na druhou stranu toho dosáhly za nízký počet iterací. *Simplex metoda* ve výkonnosti

následuje algoritmus *NLPQLP* s 35 iteracemi a Fobj = 66,14. *L-BFGS* v programech *Matlab* a *optiSLang* se podstatně liší v počtu iterací, stejně jako tomu bylo v TU1, a zde navíc také ve výsledku.

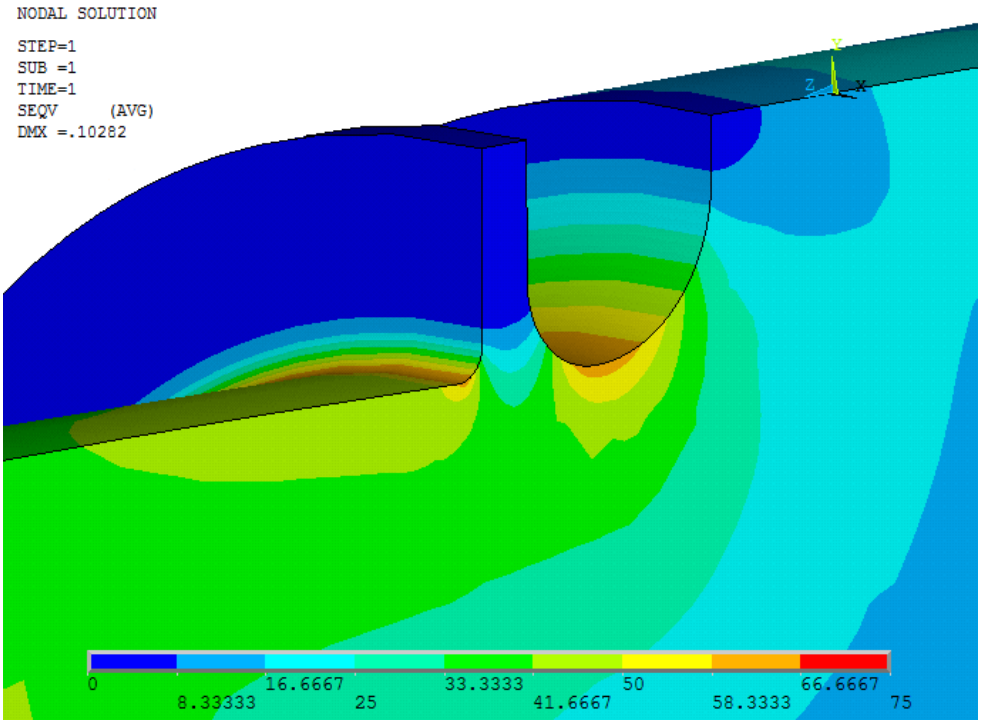
Z hlediska nalezení minima si nejlépe vedly *evoluční algoritmus* a úplně nejlépe *roj částic*, což bylo vykoupeno vysokým počtem iterací (193 a 137).

K porovnání pro tuto úlohu byly k dispozici také výsledky optimalizace přímo v programu *ANSYS*. Ty nebyly nijak výrazně dobré ani v počtu iterací, ani v přesnosti výsledku. V této testovací úloze, v rámci daného nastavení a v porovnání s ostatními výsledky v tabulce 4.6, vykazuje spíše vyšší počet iterací.



Obrázek 4.8: TU3 - počáteční stav [3 3 7 3], red.napětí HMH [MPa]

V přílohách je přiloženo řešení bez zápichu.

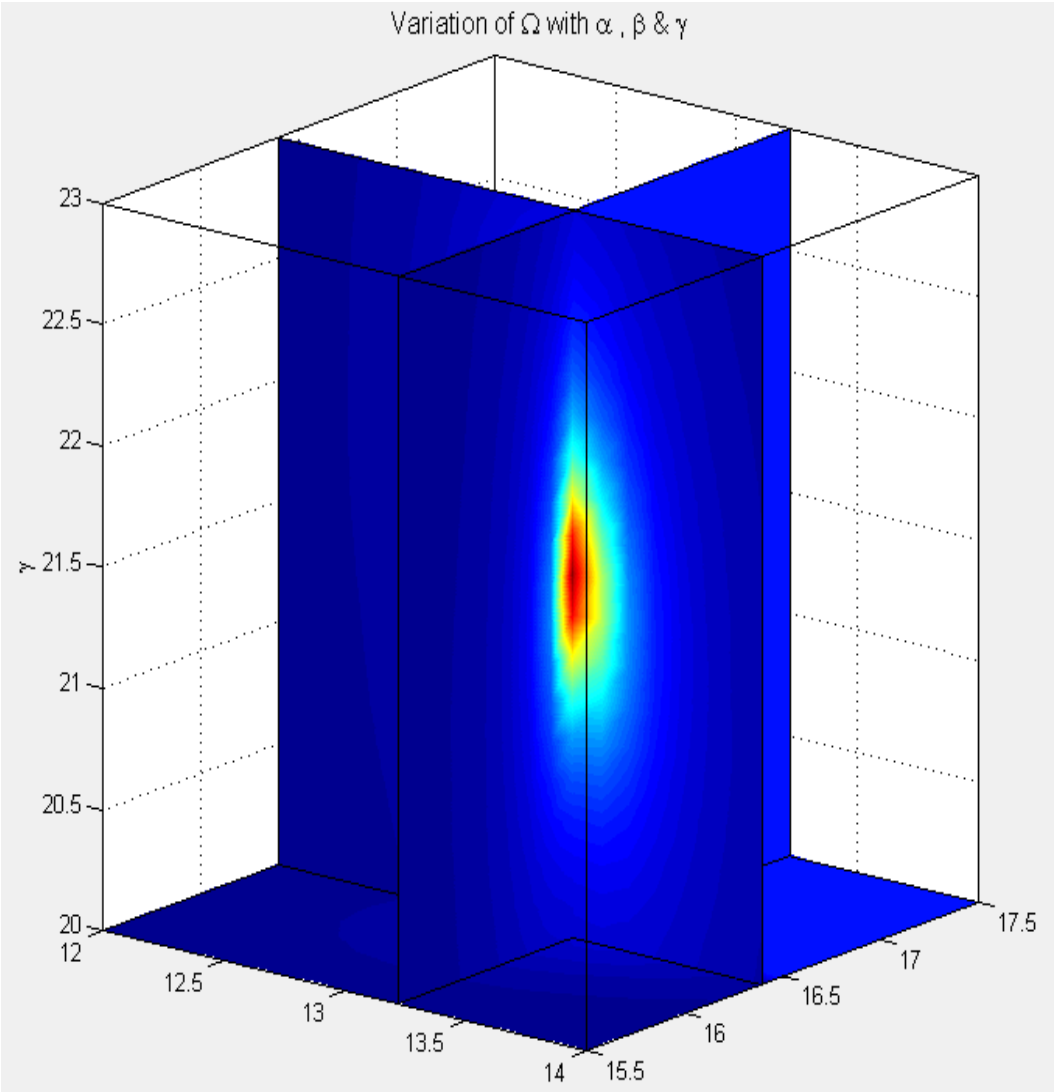


Obrázek 4.9: TU3 - nejlepší řešení roj částic [2,62 4,78 6,59 1,82], red.napětí HMH [MPa]

4.2.4. Průzkum prostoru

Vzhledem k nízkému počtu proměnných v testovacích úlohách a nízké výpočetní náročnosti úloh je možné prozkoumat chování funkce v prostoru až 3-proměnných najednou a výsledek vizualizovat. Jedná se vlastně o *metodu prohledávání v síti*, která bývá tímto způsobem využívána k prozkoumání celkového prostoru a k porozumění chování funkce. To bývá následně využito v kombinaci s další (obvykle lokální) optimalizační metodou. Tímto způsobem se můžeme podívat na chování funkce testovacích úloh a vyhodnotit tak lépe výsledky testů.

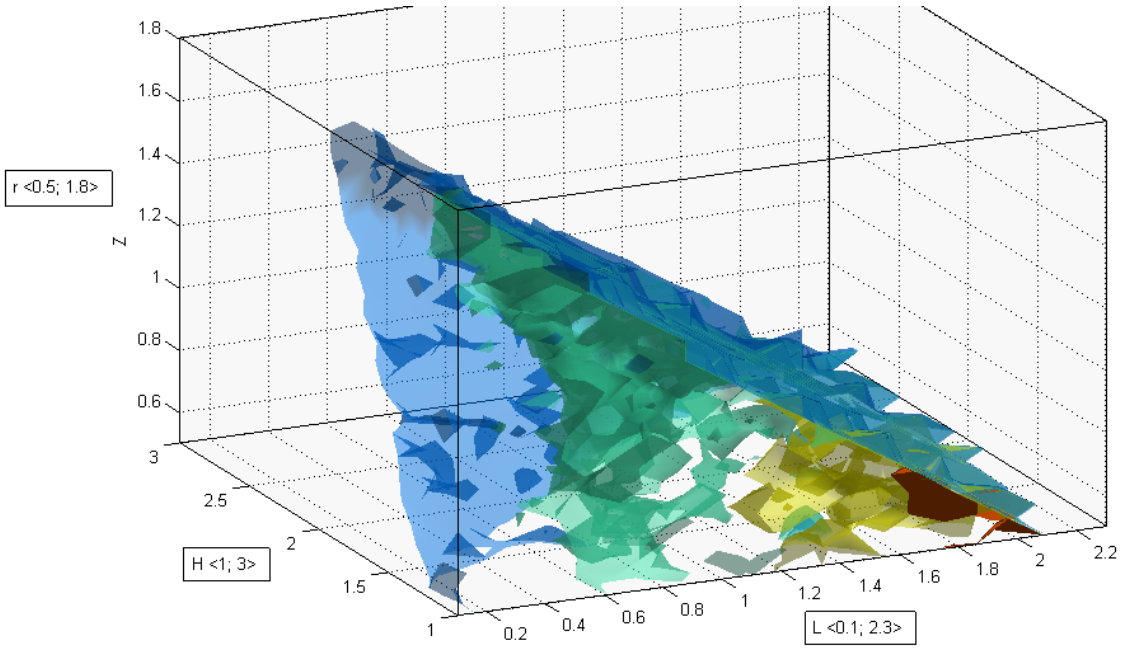
Testovací úloha č.1



Obrázek 4.10: TU1 – průzkum prostoru - řezy

Funkce má hladký průběh s jediným globálním minimem. Graf byl sestrojen v rozlišení 10x10x10, tj.1000 iterací. Pro lepší vizualizaci se barva řídí převrácenou hodnotou funkce, tj. (1/Fobj), protože zatímco blízko minima se hodnoty funkce pohybují v řádu tisícin, tak vzdálenější body nabývají hodnot v desítkách. Takové zobrazení by nevystihlo průběh v blízkosti minima.

Testovací úloha č.2

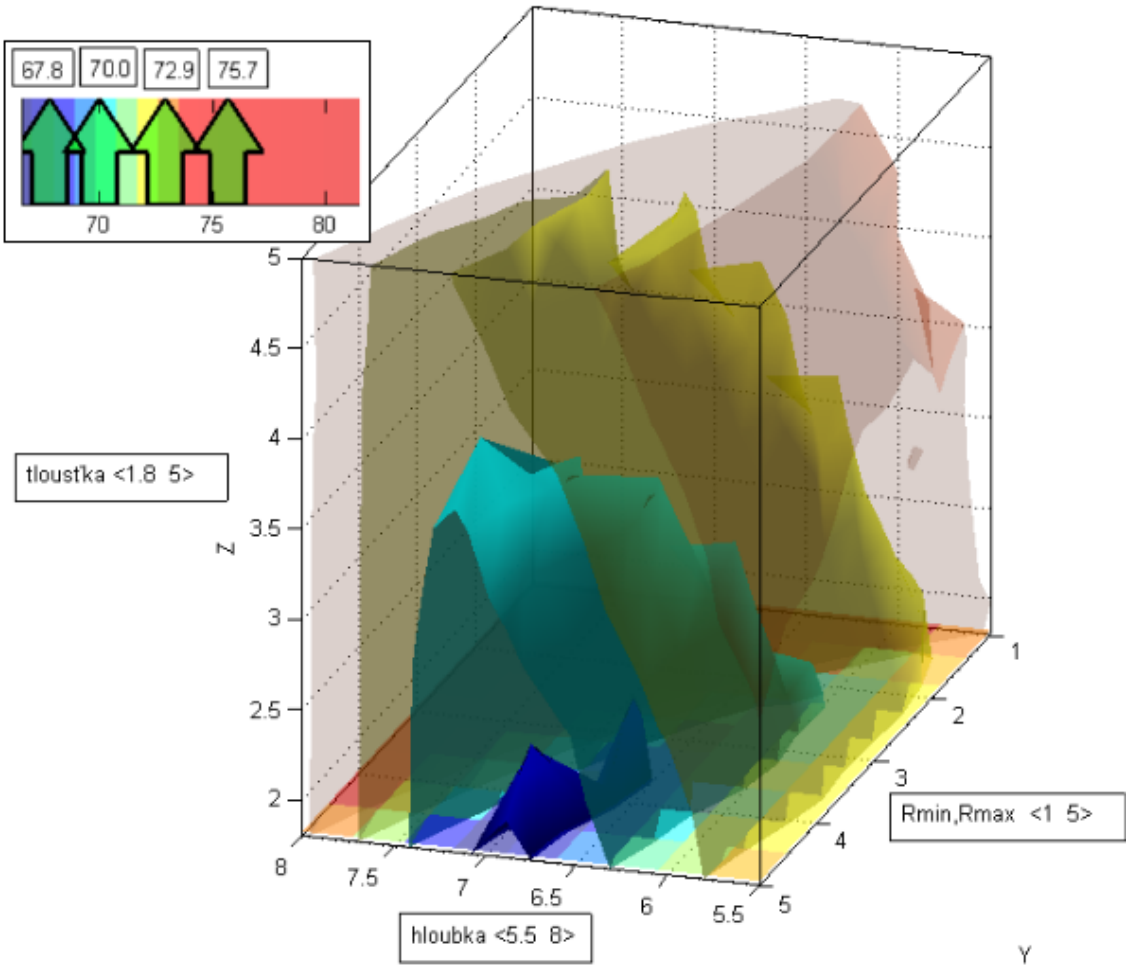


Obrázek 4.11: TU2 - průzkum oblasti. Oblast s minimem červeně

Funkce TU2 má sice jednu oblast minima, ale isopovrchy na obrázku 4.11 (tj. 3D povrchy ve kterých mají body stejnou hodnotu) jsou kostrbaté a snadno se při optimalizaci může hledání zastavit v lokálním minimu. Isopovrchy jsou tím méně hladké čím blíže skutečnému minimu jsou. Na obr. 4.11 jsou znázorněny čtyři isopovrchy, červeně je oblast ve které se nachází minimum, modrý isopovrch má naopak nejvyšší funkční hodnotu.

Graf byl sestrojen s rozlišením 21x21x21, tj.9261 funkčních vyhodnocení. Je tedy nepravděpodobné, že by byly isopovrchy deformovány nízkým rozlišením.

Testovací úloha č.3



Obrázek 4.12: TU3 – průzkum prostoru (Rmin a Rmax jsou zde prezentovány jedinou proměnnou)

TU3 má 4 parametry. Vzhledem k nízkému vlivu rozdílu mezi **Rmin** a **Rmax** budou tyto sloučeny tak, že 3D graf bude vytvořen funkcí kde bude platit **Rmin** = **Rmax** a TU3 se pro účel průzkumu prostoru stane 3-parametrická. To způsobuje mírnou odlišnost od skutečného průběhu funkce, ale pro účel průzkumu prostoru to můžeme zanedbat.

Obrázek 4.12 znázorňuje 4 isopovrchy, které přímo značí maximální redukované napětí v modelu. Modrá zobrazuje oblast minima., kde vidíme, že blízko globálního minima se nachází více než jedno lokální minimum. To je důvod proč některé algoritmy nedosáhly minima (okolo hodnoty 65), ale zastavili se na hodnotě okolo 70 MPa.

4.3. Celkové vyhodnocení testování

V rámci testovaných úloh se jako nejspolehlivější jeví *metoda Simplex*. Ta sice konverguje pomaleji, ale zase si poradila se všemi úlohami včetně TU2.

Z gradientních metod se osvědčil nejvíce *NLPQLP*, což je algoritmus založený na Newtonovském principu, stejně jako *kvazi-Newtonova metoda*. *NLPQLP* metoda je robustnější než *BFGS* a *L-BFGS* právě proto, že zohledňuje vzniklé nelinearity. Rychlost konvergence je však srovnatelná s ostatními testovanými gradientními metodami.

O něco hůře dopadla *Cauchyho metoda* s ohledem na rychlost i schopnost nalezení přesného řešení. To mohlo být způsobeno nevhodným nastavením volitelných parametrů (přesnost, délka kroku) či nastavením metody 1D optimalizace, kterou využívá (zde použita *metoda Zlatého řezu* a *kvadratická interpolace*). Stejně tak mohl být výkon algoritmu ovlivněn implementací omezujících podmínek (metoda byla původně napsána pro neomezené hledání a tvůrce práce do ní implementoval omezení na intervalu).

Srovnávací stochastické metody *roj částic* a *evoluční algoritmus* konvergovaly mnohem pomaleji než *Simplex metoda* či *gradientní metody* a navíc ve svém základním nastavení neposkytly podporu ani v optimalizaci špatně konvergující testovací úlohy č.2. Jejich robustnost se alespoň kladně projevila v TU3 kdy obě metody dosáhly nejpřesnějších výsledků (ačkoliv opět za delší výpočetní čas).

Jako doplněk testování byl k dispozici výpočet gradientní metody z programu ANSYS/Mechanical pro TU3. Jeho výsledky lze hodnotit spíše jako slabší (vyšší počet iterací, nezkonvergoval až k hodnotě 65).

	TU1		TU2		TU3	
	poč. iter.	Fobj	poč. iter.	Fobj	poč. iter.	Fobj
Počáteční stav	1	2,07	1	4,79e-1	1	74,76
Simplex Nelder-Mead	60	3,09e-4	8	4,67e-1	21	65,82
Cauchy – zlatý řez	93	7,94e-3	nekonv.		11	70,07
Cauchy – interpolace	99	9,40e-4	nekonv.		14*	70,98
Matlab - BFGS	96	8,58e-7	nekonv.		24	69,82
Matlab - LBFGS	68	6,71e-7	nekonv.		27	69,34
optiSLang - LBFGS	103	5,43e-7	nekonv.		121	66,06
optiSLang - NLPQLP	87	4,48e-7	23	4,73e-1	35	66,14
optiSLang – evoluční	200	4,46e-3	nekonv.		193	65,04
optiSLang – roj částic	394	1,34e-5	nekonv.		137	64,75
ANSYS	--	--	--	--	50	68,22

Tabulka 4.7: TU1-TU3 - souhrn výsledků testování

	Úloha 1 **nosník konstantní pevnosti**	**Úloha 2** **koncentrace napětí v nalisovaném spoji**	**Úloha 3** **vrubové napětí na hřídeli**
Dimenze modelu	2D	2D	3D
Dimenze optimalizace	3 parametry	3 parametry	4 parametry
Meze parametrů	W1-W3: <8,50>	Vzdálenost <1 20*> Hloubka: <0,1 20*> Poloměr: <0,5 5*>	Rmin <1; 5> Rmax <1; 5> hl <5,5; 8> t <1,8; 5>
Další omezení	---	$Mk_{přen} > Mk_{jm}$	---
Cíl optimalizace	σ_y = konst. po celé délce	σ = konst. na kontaktní ploše hřídele	Minimální σ v modelu
Tvar objekt. funkce	$\sum_{i=1}^{n} (\sigma_{yi} - \sigma_{konst})^2$	σ_{max}-σ_{min} na kontaktní ploše hřídele	σ_{min}
Počáteční hodnoty	<10 10 10>	<1 1 1>	<3 3 7 3>

Tabulka 4.8: Shrnutí nejdůležitějších parametrů testovaných úloh

*** -** značí že v tomto směru není parametr omezen jinak než geometrií tělesa

5. Závěr

Práce se zabývala porovnáním optimalizačních algoritmů v tvarové parametrické optimalizaci lineárních statických strukturálních úloh. Součástí práce bylo také porovnání uživatelsky vytvořených algoritmů v prostředí programu Matlab s algoritmy v komerčních programech. Byla provedena rešerše optimalizačních metod a rešerše komerčních optimalizačních programů. Následně bylo na vybraných úlohách testováno několik vybraných algoritmů z obou skupin. Na závěr byly zhodnoceny výsledky testování a zodpovězeny otázky, které byly položeny v úvodu práce.

V první části byla provedena rešerše matematických metod parametrické optimalizace. Zaměření v této části práce bylo na metody s uplatněním při konstrukci strojních částí. Byly zkoumány *jedno-dimenzionální metody*, jakožto základ některých *více-dimenzionálních metod*, dále *metoda Simplex*, *gradientní metody* a nejpoužívanější stochastické metody *genetický algoritmus* a *roj částic*. Z těchto metod byly vybráni zástupci různých kategorií pro účel testování a byl pro ně připraven algoritmus v programu *Matlab,* případně bylo využito již implementovaných algoritmů (*BFGS* a *L-BFGS*). Testovány byly *Simplex metoda*, gradientní *Cauchyho metoda* a *kvazi-Newtonova metoda (*v podobě algoritmu *BFGS* a *L-BFGS)*.

V další části práce byl proveden průzkum komerčních optimalizačních programů, které jsou využívány ve strukturálních analýzách v kombinaci s MKP programem. Z nich byl jako zástupce pro testování vybrán program *optiSLang*, jehož matematické nástroje byly podrobněji popsány. K testování pak byly vybrány algoritmy podobné těm dostupným v programu *Matlab* (*BFGS*). K porovnání výkonnosti s jinými metodami než těmi, které je uživatel schopen si poměrně rychle připravit sám (*Simplex metoda*, *gradientní metody*), byly vybrány také dvě stochastické metody, aby bylo možné posoudit výkonnost testovaných algoritmů i vůči jiným metodám.

V části porovnávání, výkonnosti algoritmů na testovacích úlohách, byly vybrány tři úlohy s malým počtem proměnných (3-4). Na těch byla provedena optimalizace pomocí uvedených algoritmů v kombinaci s MKP programem ANSYS/Mechanical. Výsledky byly shrnuty na konci kapitoly 4.

Ladění algoritmů na jejich nejlepší efektivitu by byla časově náročná operace a

autor práce se tímto nezabýval. Proto porovnávání algoritmů za jiných podmínek může vést k odlišným výsledkům.

Doporučení

Bylo zjištěno, že komerční matematické nástroje určené pro inženýrskou praxi, jehož zástupcem zde byl program optiSLang, mají velice dobře implementované matematické metody. S ohledem na čas tvorby vlastních algoritmů a cenu MKP programů je výhodnější použít zakoupený optimalizační program a snížit tím časové a finanční náklady.

Pokud by bylo potřeba ušetřit na nákupu optimalizačního programu, je možné vytvořit si vlastní algoritmus, eventuálně upravit volně dostupné kódy a použít je v některém z volně dostupných matematických nástrojů, jako například Octave (OS alternativa k programu Matlab).

Dobrým kompromisem mezi robustností a výkonem je metoda Simplex, avšak inženýr brzo narazí na úlohy, kde bude potřeba využít jiných přístupů. Typickým postupem bývá nejprve průzkum prostoru nebo citlivostní analýza, aby mohlo být analyzováno chování úlohy. Následně je aplikován optimalizační algoritmus, nejlépe jejich kombinace, která zajistí efektivní nalezení globálního minima s následným vyladěním v lokální oblasti prostoru proměnných parametrů.

Návrh na další zkoumání

Tato práce poskytuje pouze základní přehled o dostupných metodách dané problematiky a testování je provedeno na omezeném počtu úloh. Nebyl také věnován čas nejlepšímu naladění algoritmů. Další zkoumání navazující na tuto práci by se proto mohlo zabývat:

- Studií charakteristik nejběžnějších inženýrských úloh s velkým počtem reálně existujících problémů.
- Vlivem nastavení volitelných parametrů jednotlivých algoritmů na jejich výkonnost.

- Rozšíření o další metody parametrické optimalizace, které zde nebyly testovány, například o ty popsané v kapitole 3.2.1.

6. Seznam použité literatury

[1] MAREŠ, Tomáš. *Konstrukční optimalizace*. Vyd. 1. Praha : Nakladatelství ČVUT, 2007. 106 s. ISBN 978-80-01-03696-9. [kniha]

[2] YANG, Won Young, et al. *Applied Numerical Methods Using MATLAB* [online]. New Jersey : John Wiley & Sons, 27 JAN 2005 [cit. 2010-12-07]. Dostupné z WWW: <http://onlinelibrary.wiley.com/book/10.1002/0471705195>. ISBN 9780471705192.

[3] RAVINDRAN, A.; RAGSDELL, K. M.; REKLAITIS, G. V. *Engineering Optimization : Methods and Applications* [online]. Second Edition. New Jersey : John Wiley & Sons, 23 MAR 2007 [cit. 2010-12-07]. Dostupné z WWW: <http://onlinelibrary.wiley.com/book/10.1002/9780470117811>. ISBN 9780470117811.

[4] RAO, Singiresu S. *Engineering Optimization Theory and Practice : Fourth Edition*. New Jersey : John Wiley & Sons, Inc., Hoboken, 2009. 840 s. ISBN 978-0-470-18352-6. [kniha]

[5] ANSYS, Inc. *Introduction to ANSYS Workbench Scripting in ANSYS 12.1* [online]. [s.l.] : [s.n.], 2009 [cit. 2011-01-12]. Dostupné z WWW: <http://files.engineering.com/download.aspx?folder=87f33752-ae15-4e43-8c21-686020b5e82e&file=Workbench_Scripting.pdf>. [e-kniha]

[6] ANSYS, Inc. *ANSYS Workbench Scripting Guide* [online]. [s.l.] : SAS IP, Inc., 2009 [cit. 2011-01-12]. Dostupné z WWW: <www1.ansys.com/customer/content/documentation/121/wb2_js.pdf>. [e-kniha]

[7] GIORGI, G.; GUERRAGGIO, A.; THIERFELDER, J. *Mathematics of Optimization : Smooth and Nonsmooth Case* [online]. [s.l.] : Elsevier, 28 March 2004 [cit. 2010-12-15]. Dostupné z WWW: <http://www.sciencedirect.com/science/book/9780444505507>. ISBN 978-0-444-50550-7.

[8] Dynardo, Dynamic Software and Engineering GmbH. *OPTISLANG - THE OPTIMIZING STRUCTURAL LANGUAGE : Version 3.1.4*. Weimar, Luthergasse 1D, Germany : Dynardo, 2010. 809 s.

[9] SÁGA, Milan, et al. Aplikácia optimalizačných algoritmov v mechanike telies. VTS pri ŽU : Žilina, 2006. 240 s. ISBN 80-969165-9-9.

[10] LENERT, Jiří. Pružnost a pevnost II. Ostrava : VŠB-TUO, 1998. 173 s. ISBN 0-7078-572-1.

[11] HERTZMANN, Aaron ; FLEET, David . Quadratics [online]. Toronto : University of Toronto, 2009 [cit. 2011-05-12]. Dostupné z WWW: <http://www.cs.toronto.edu/~fleet/courses/D11/fall10/notesReadings.html>. [e-kniha]

[12] *FE Design* [online]. 2011 [cit. 2011-05-20]. TOSCA Structure. Dostupné z

WWW: <http://www.fe-design.de/en/products/tosca-structure/>.

[13] *Vanderplaats Research & Development, Inc.* [online]. 2011 [cit. 2011-05-19]. Genesis Structural Analysis and Optimization Software. Dostupné z WWW: <http://www.vrand.com/Genesis.html>. [webová stránka]

[14] *DS SIMULIA* [online]. 2011 [cit. 2011-05-20]. Isight. Dostupné z WWW: <http://www.simulia.com/products/isight.html>.

[15] *FEA-Opt Technology* [online]. 2011 [cit. 2011-05-20]. SmartDO. Dostupné z WWW: <http://www.fea-optimization.com/optm/SmartDO_e.htm>.

[16] *DYNARDO GMBH* [online]. 2011 [cit. 2011-05-20]. Dostupné z WWW: <http://www.dynardo.com/>.

[17] *Charles Nippert* [online]. 14.12.2005 [cit. 2011-05-22]. Simplex Optimization. Dostupné z WWW: <quantum.soe.widener.edu:280/E616/SimplexOptimization.doc>. [webová stránka]

Použitý software

- ANSYS release 13.0
- Matlab 7.10 (R2010a) od MathWorks
- OptiSLang od Dynardo GmbH – laskavě zapůjčený firmou SVS FEM s.r.o.
- ostatní použitý software použitý k vytvoření diplomové práce: LibreOffice, Inkscape, Gimp, PDF-Xchange Viewer

7. Seznamy tabulek a obrázků

Seznam tabulek

Tabulka 2.1: porovnání eliminačních metod [4].....20
Tabulka 4.1: TU1 – vlastnosti úlohy.....51
Tabulka 4.2: TU1 – souhrn výsledků testování.....52
Tabulka 4.3: TU2 – vlastnosti úlohy.....55
Tabulka 4.4: TU2 – souhrn výsledků testování.....55
Tabulka 4.5: TU2 – vlastnosti úlohy.....58
Tabulka 4.6: TU3 – souhrn výsledků testování.....59
Tabulka 4.7: TU1-TU3 - souhrn výsledků testování.....66
Tabulka 4.8: Shrnutí nejdůležitějších parametrů testovaných úloh.....66

Seznam obrázků

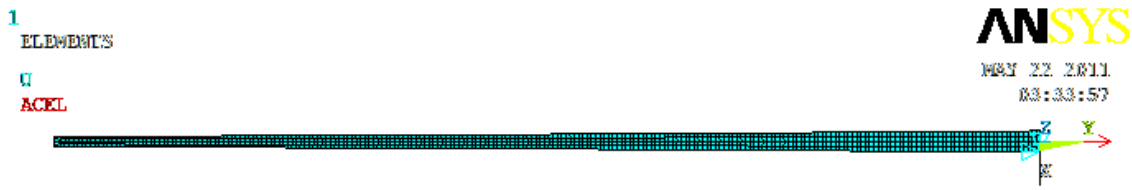
Obrázek 2.1: Dělení jedno-dimenzionálních optimalizačních metod [4].....16
Obrázek 2.2: ukázka unimodálních funkcí [4].....17
Obrázek 2.3: umístění bodů x1-x6 v iteračním procesu při Fibonacciho hledání n = 6 na intervalu <0, 3>.....19
Obrázek 2.4: průběh iterací u Newtonovy metody.....25
Obrázek 2.5: Průběh iterací metody sečen, znázornění parametru s.....27
Obrázek 2.6: Metoda prohledávání v síti.....29
Obrázek 2.7: Metoda předlohového hledání. Přerušované čáry vyznačují řídící směry pro další iterace, které metoda využívá.[4].....30
Obrázek 2.8: Powellova metoda. Znázornění iteračního procesu a determinace následně použitého směru.....32
Obrázek 2.9: Simplex metoda v 2D optimalizaci.....33
Obrázek 2.10: Typický průběh evolučního algoritmu [8].....37
Obrázek 3.1: optiSLang - prostředí - nabídka.....41
Obrázek 4.1: Testovací úloha č.1 - nákres.....50
Obrázek 4.2: TU1 - počáteční stav nosníku s parametry [10 10 10] (vpravo uchycení,

viz SS z obr.4.1), osové napětí [MPa].....52
Obrázek 4.3: TU1 - nejlepší řešení s hodnotami W1-W3 [13,01 16,76 21,60] (optiSLang - NLPQLP), osové napětí [MPa].....52
Obrázek 4.4: Testovací úloha č.2 - nákres.....53
Obrázek 4.5: TU2 - počáteční stav [1 1 1], red.napětí HMH [MPa].....56
Obrázek 4.6: TU2 – nejlepší nalezené řešení [1 1,69 0,75], optiSLang – roj částic, red.napětí HMH [MPa].....57
Obrázek 4.7: Testovací úloha č.3 - nákres.....57
Obrázek 4.8: TU3 - počáteční stav [3 3 7 3], red.napětí HMH [MPa].....60
Obrázek 4.9: TU3 - nejlepší řešení roj částic [2,62 4,78 6,59 1,82], red.napětí HMH [MPa].....61
Obrázek 4.10: TU1 – průzkum prostoru - řezy.....62
Obrázek 4.11: TU2 - průzkum oblasti. Oblast s minimem červeně.....63
Obrázek 4.12: TU3 – průzkum prostoru (Rmin a Rmax jsou zde prezentovány jedinou proměnnou).....64
Obrázek 8.1: TU1 - MKP model, cca 1100 uzlů.....75
Obrázek 8.2: TU2 - MKP model, cca 1200 uzlů.....75
Obrázek 8.3: TU3 - MKP model, cca 51 000 uzlů.....75
Obrázek 8.4: TU2 - výpočet bez zápichu (umístěn daleko), Fobj = 5,66e-1, red.napětí HMH [MPa].....76
Obrázek 8.5: TU3 - výpočet bez zápichu (umístěn daleko), red.napětí HMH [MPa]….76
Obrázek 8.6: TU1 – Simplex Nelder-Mead, Fobj = 3,09e-4 za 60 iterací.....78
Obrázek 8.7: TU1 - Cauchy - Zlatý řez, Fobj = 7,94e-3 za 93 iterací.....78
Obrázek 8.8: TU1 - Cauchy - interpolace, Fobj = 9,40e-4 za 99 iterací.....79
Obrázek 8.9: TU1 - BFGS - Matlab, Fobj = 8,58e-7 za 96 iterací.....79
Obrázek 8.10: TU1 - LBFGS - Matlab, Fobj = 6,71e-7 za 68 iterací.....80
Obrázek 8.11: TU1 – LBFGS – optiSLang, Fobj = 5,43e-7 za 103 iterací.....80
Obrázek 8.12: TU1 – NLPQLP – optiSLang, Fobj = 4,48e-7 za 87 iterací.....81
Obrázek 8.13: TU1 – Evoluční algoritmus Fobj = 4,46e-3 za 200 iterací. 81
Obrázek 8.14: TU1 – Roj částic Fobj = 1,34e-5 za 394 iterací. 82
Obrázek 8.15: TU2 - simplex Nelder-Mead, Fobj = 4,77e-1 za 8 iterací.....82
Obrázek 8.16: TU2 - NLPQLP – optiSLang, Fobj = 4,73e-1 za 23 iterací.....83

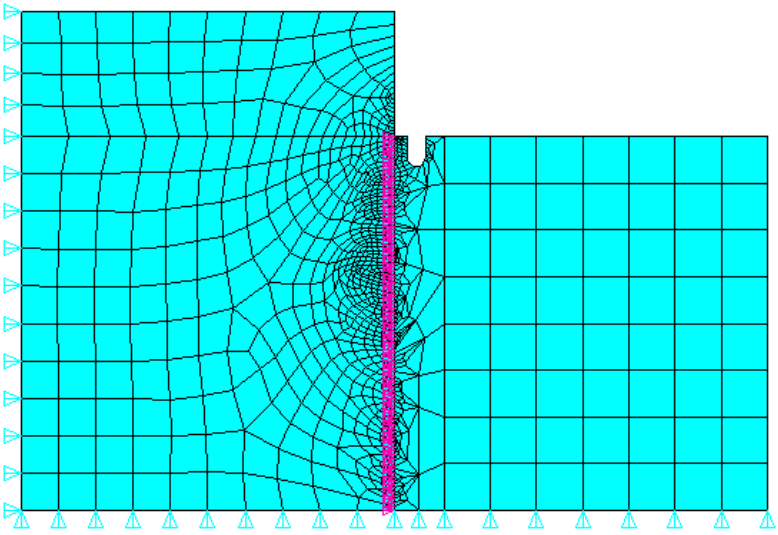
Obrázek 8.17: TU2 - simplex Nelder-Mead, přepočet pro bod [3 1 1]. Fobj = 4,83e-1 za 22 iterací z bodu [3 1 1] kde Fobj = 5,34e-1.....83
Obrázek 8.18: TU2 - NLPQLP, nekonvergující přepočet pro bod [3 1 1].....84
Obrázek 8.19: TU3 - simplex Nelder-Mead, Fobj = 65,82 za 21 iterací.....84
Obrázek 8.20: TU3 - Cauchy - Zlatý řez, Fobj = 70,07 za 11 iterací.....85
Obrázek 8.21: TU3 - Cauchy - interpolace, Fobj = 70,98 za 14 iterací (výpočet havaroval po 18ti iteracích z důvodu chyby v APDL souboru).....85
Obrázek 8.22: TU3 - BFGS - Matlab, Fobj = 69,57 za 24 iterací.....86
Obrázek 8.23: TU3 - LBFGS - Matlab, Fobj = 69,34 za 27 iterací.....86
Obrázek 8.24: TU3 - LBFGS – optiSLang, Fobj = 66,06 za 121 iterací.....87
Obrázek 8.25: TU3 - NLPQLP – optiSLang, Fobj = 6,61e-1 za 36 iterací.....87
Obrázek 8.26: TU3 - Evoluční algoritmus, Fobj = 65,04 za 200 iterací.....88
Obrázek 8.27: TU3 - Roj částic, Fobj = 64,75 za 137 iterací.....88

8. Přílohy

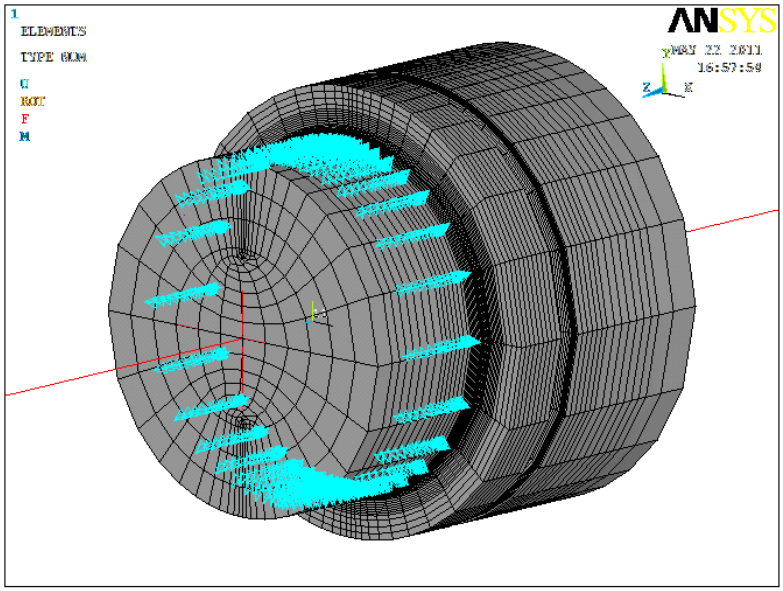
8.1. Příloha 1 - Modely testovacích úloh a výpočty bez vrubu



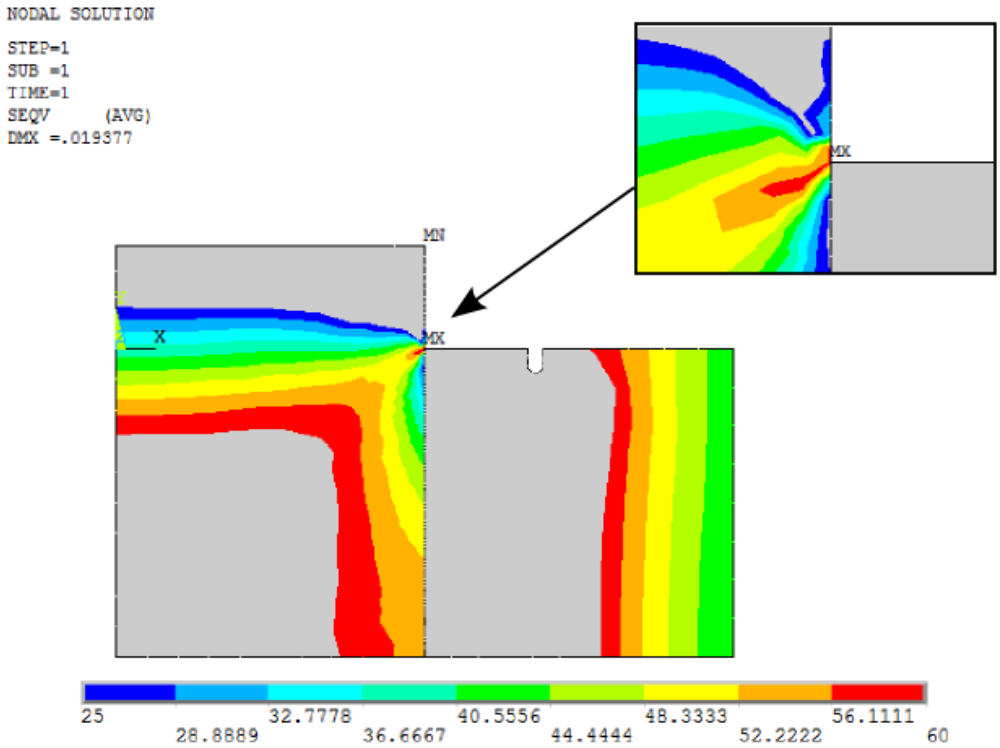
Obrázek 8.1: TU1 - MKP model, cca 1100 uzlů



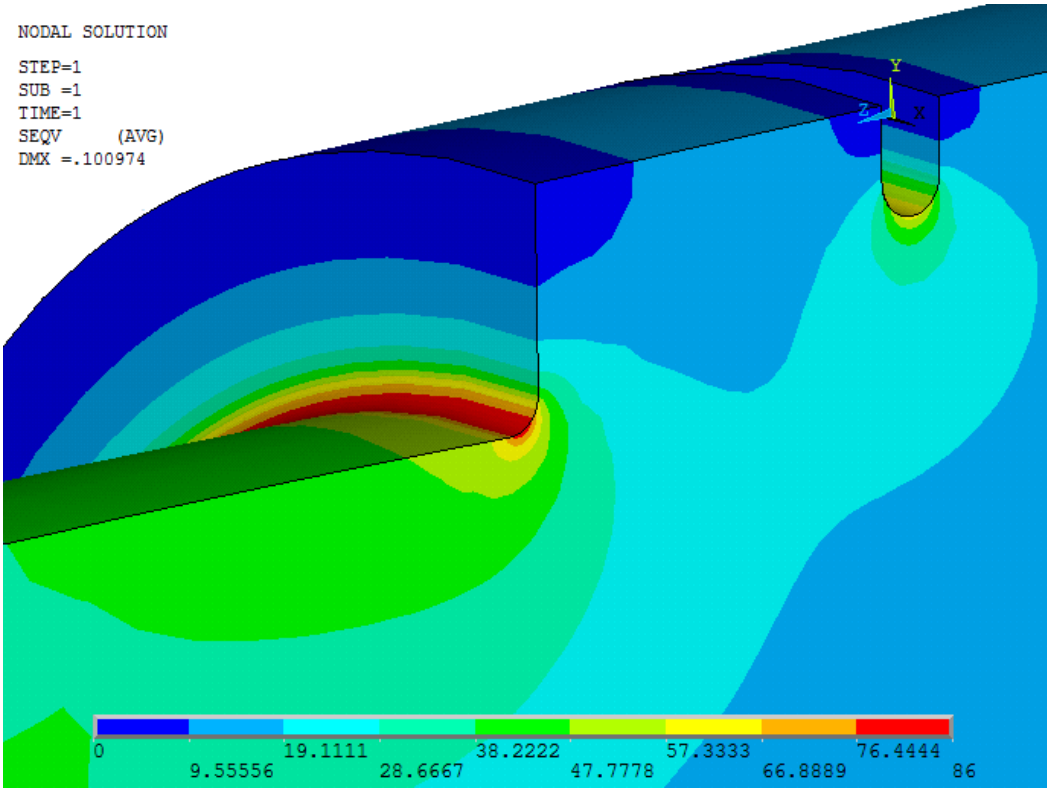
Obrázek 8.2: TU2 - MKP model, cca 1200 uzlů



Obrázek 8.3: TU3 - MKP model, cca 51 000 uzlů



Obrázek 8.4: TU2 - výpočet bez zápichu (umístěn daleko), Fobj = 5,66e-1, red.napětí HMH [MPa]



Obrázek 8.5: TU3 - výpočet bez zápichu (umístěn daleko), red.napětí HMH [MPa]

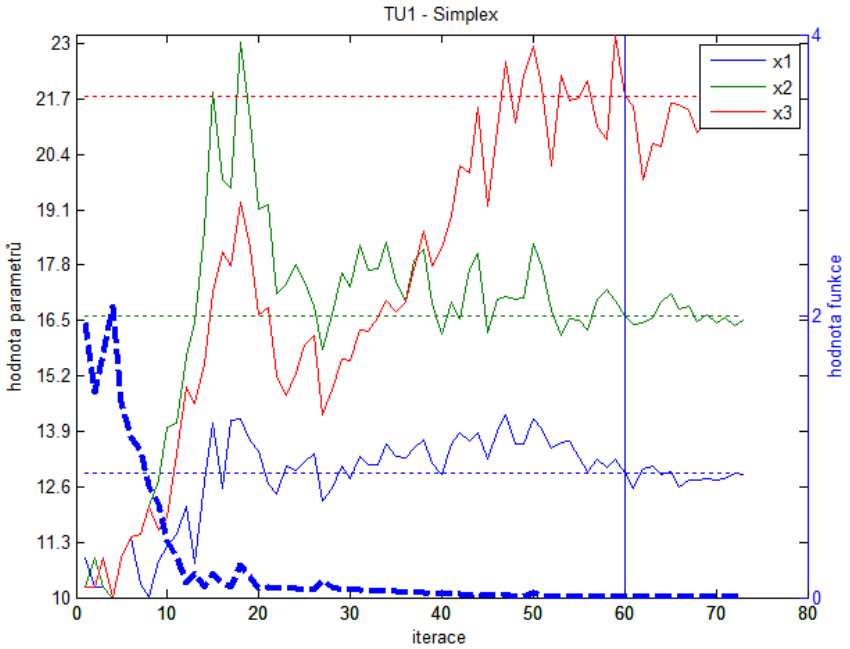
8.2. Příloha 2 - Průběhy objektivní funkce pro provedenou optimalizaci testovacích úloh

Vysvětlivky ke grafům

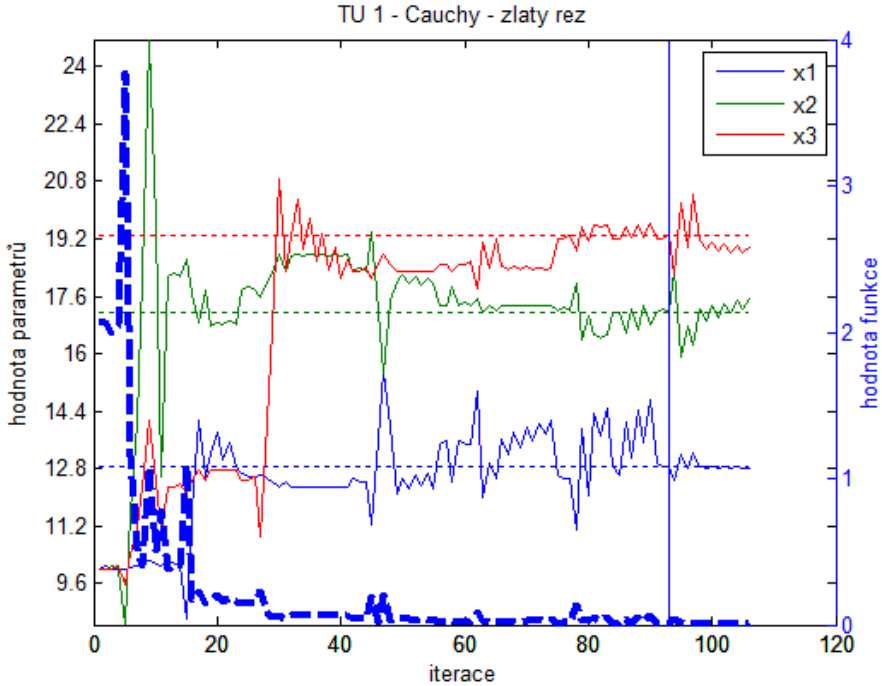
Grafy z programu Matlab (ty bez názvu optiSLang) jsou kombinované grafy hodnoty objektivní funkce a parametrizovaných proměnných. Na levé straně mají osu pro hodnoty proměnných, na pravé straně osu pro hodnotu objektivní funkce. Průběh objektivní funkce je znázorněn tlustou přerušovanou čárou. Tenké zleva doprava (3 nebo 4) jsou průběhy proměnných. Pro dobrou vizualizaci minima grafy obsahují také porovnávací rovné čáry. Přerušované vodorovné čáry znázorňují hodnotu nejlepšího výsledku. Svislá modrá čára pak zobrazuje počet iterací za který byl nejlepší výsledek nalezen což může a nemusí být v poslední iteraci.

Průběhy objektivní funkce z programu *optiSLang* jsou na grafech mírně zavádějící. Ukazují totiž počet navržených design bodů a ten se někdy neshoduje s počtem skutečných vyhodnocení funkce. Pokud jsou totiž design body totožné či téměř totožné s již existujícím designem tak se funkce znovu nevyhodnocuje. Skutečný počet vyhodnocení funkce je vyčten z výstupového souboru a je doplněn v textu pod obrázkem. U některých grafů z *optiSLangu* byly také změněny rozsahy osy *y* z důvodu lepší rozlišitelnosti menších funkčních hodnot na úkor nezobrazení některých „úletů“ funkčních hodnot, obzvláště v počátečních iteracích.

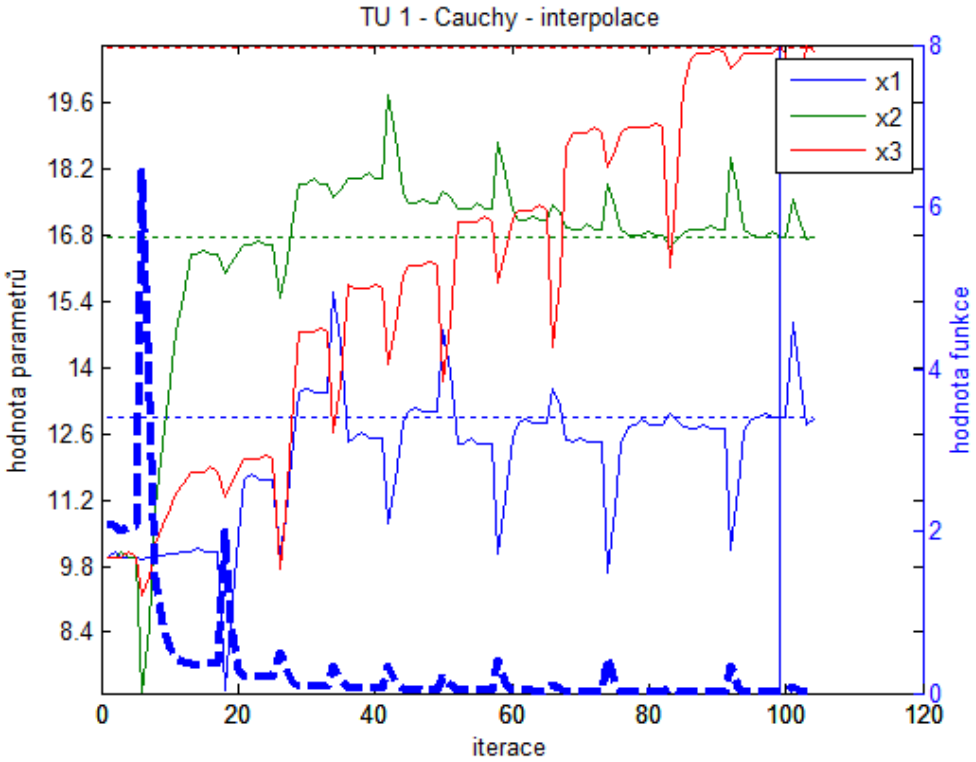
8.2.1. Průběh iterací testovací úlohy č.1



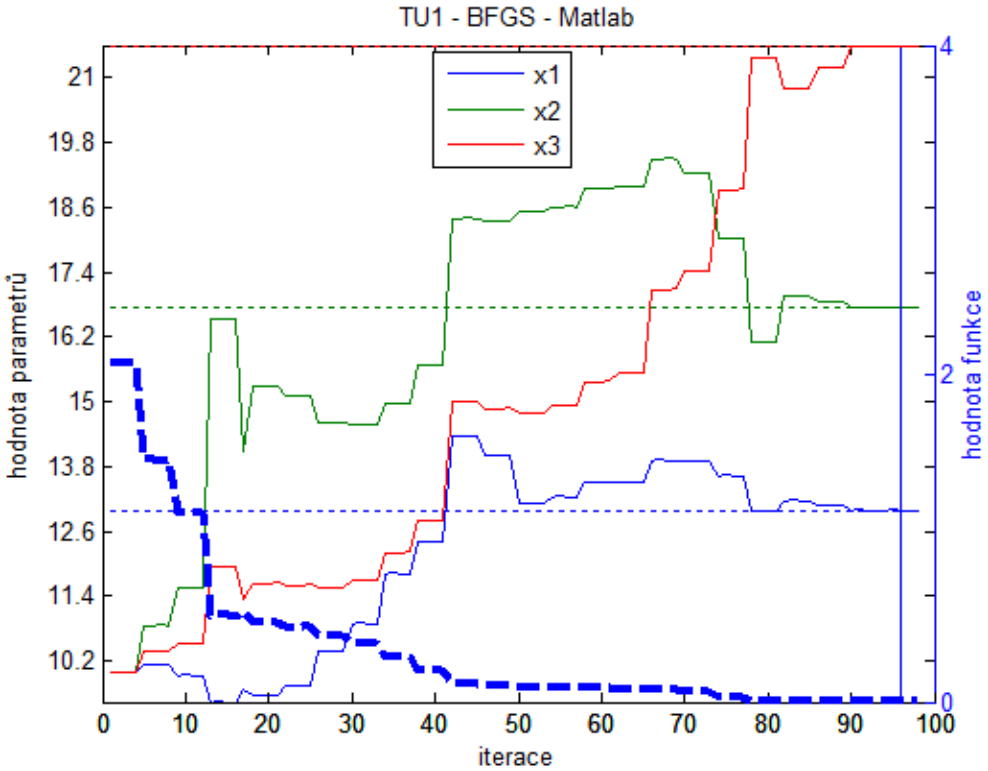
Obrázek 8.6: TU1 – Simplex Nelder-Mead, Fobj = 3,09e-4 za 60 iterací



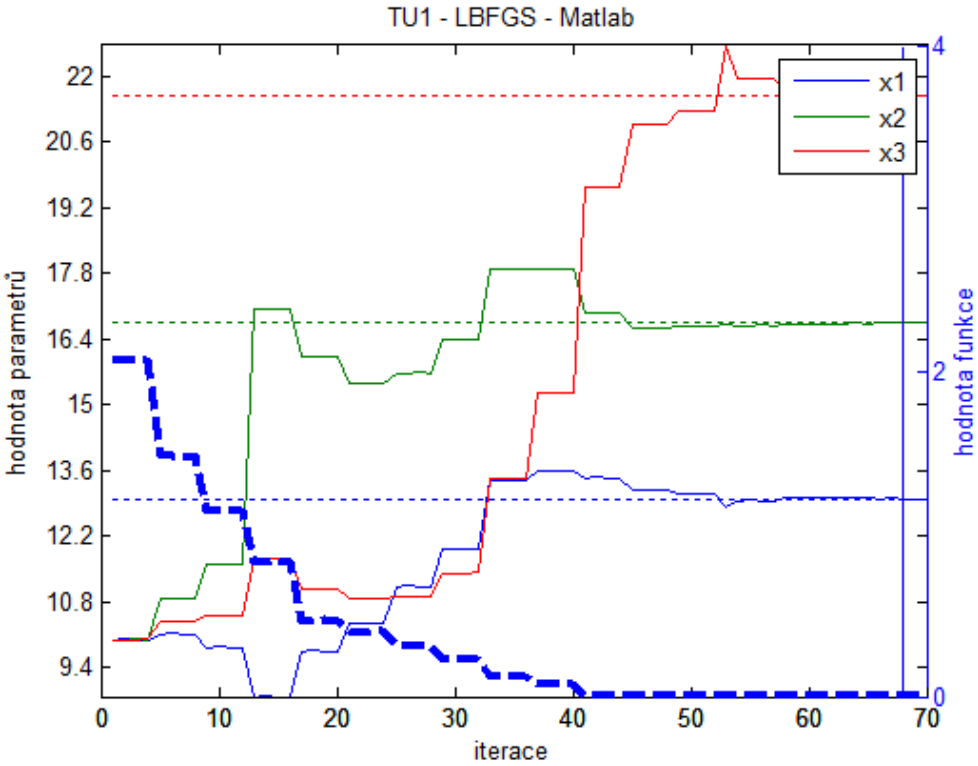
Obrázek 8.7: TU1 - Cauchy - Zlatý řez, Fobj = 7,94e-3 za 93 iterací



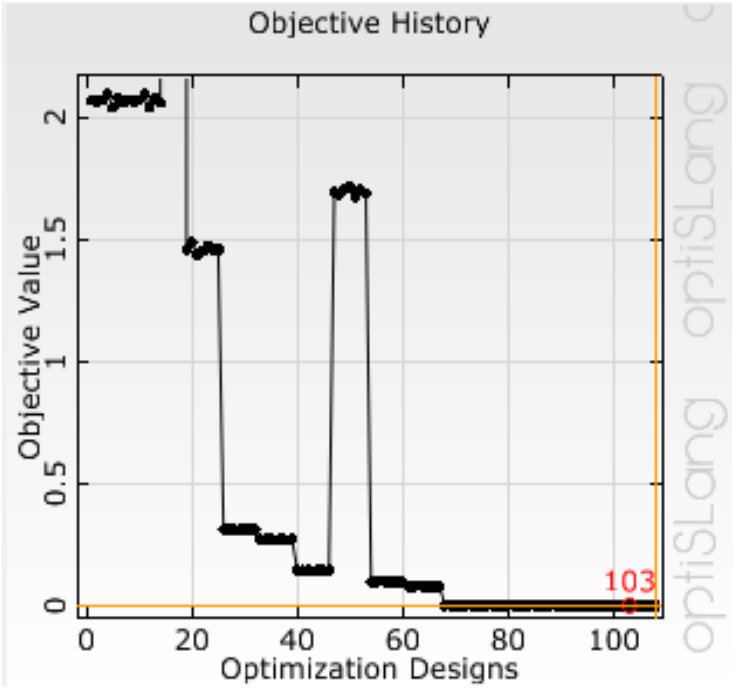
Obrázek 8.8: TU1 - Cauchy - interpolace, Fobj = 9,40e-4 za 99 iterací



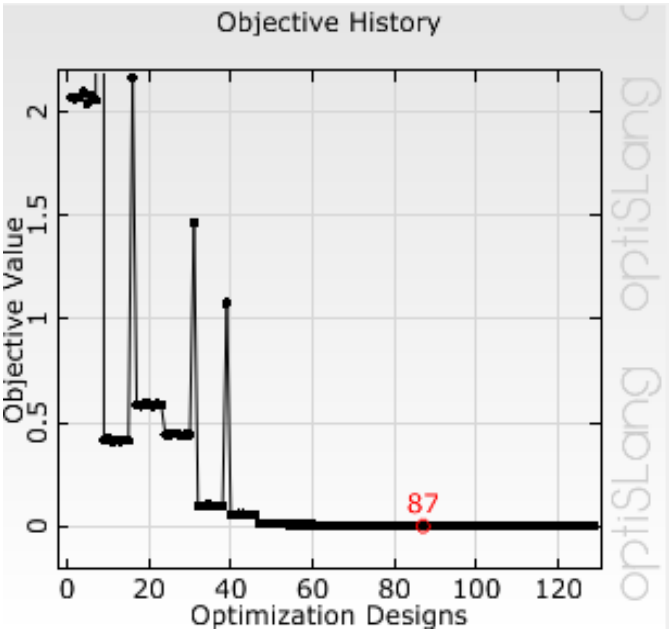
Obrázek 8.9: TU1 - BFGS - Matlab, Fobj = 8,58e-7 za 96 iterací.



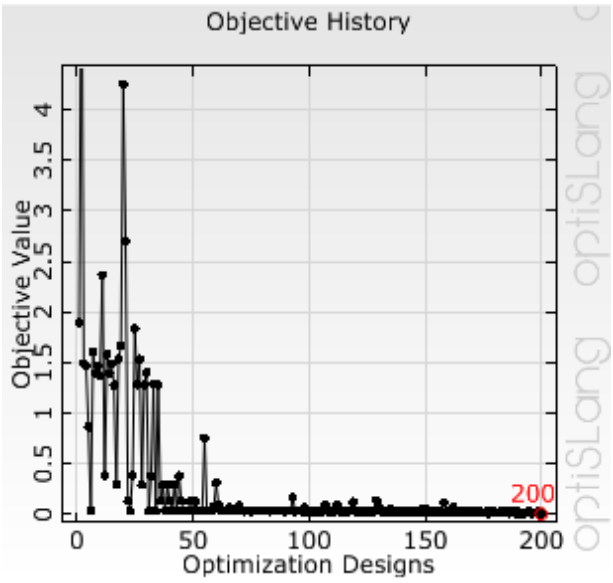
Obrázek 8.10: TU1 - LBFGS - Matlab, Fobj = 6,71e-7 za 68 iterací.



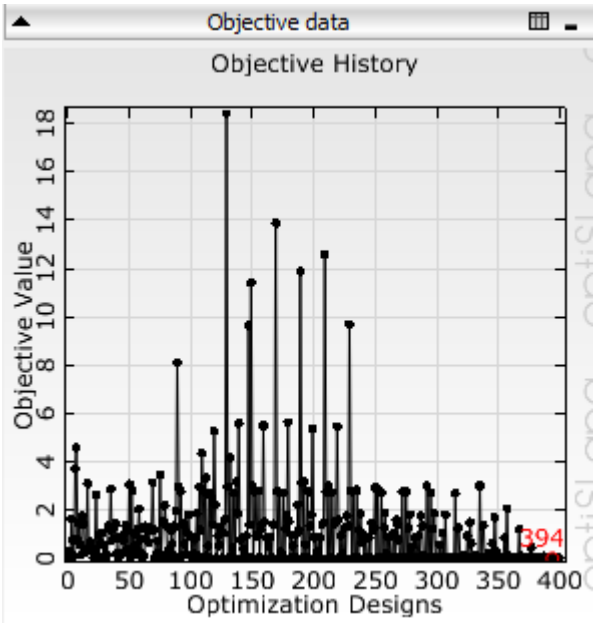
Obrázek 8.11: TU1 – LBFGS – optiSLang, Fobj = 5,43e-7 za 103 iterací.



Obrázek 8.12: TU1 – NLPQLP – optiSLang, Fobj = 4,48e-7 za 87 iterací.

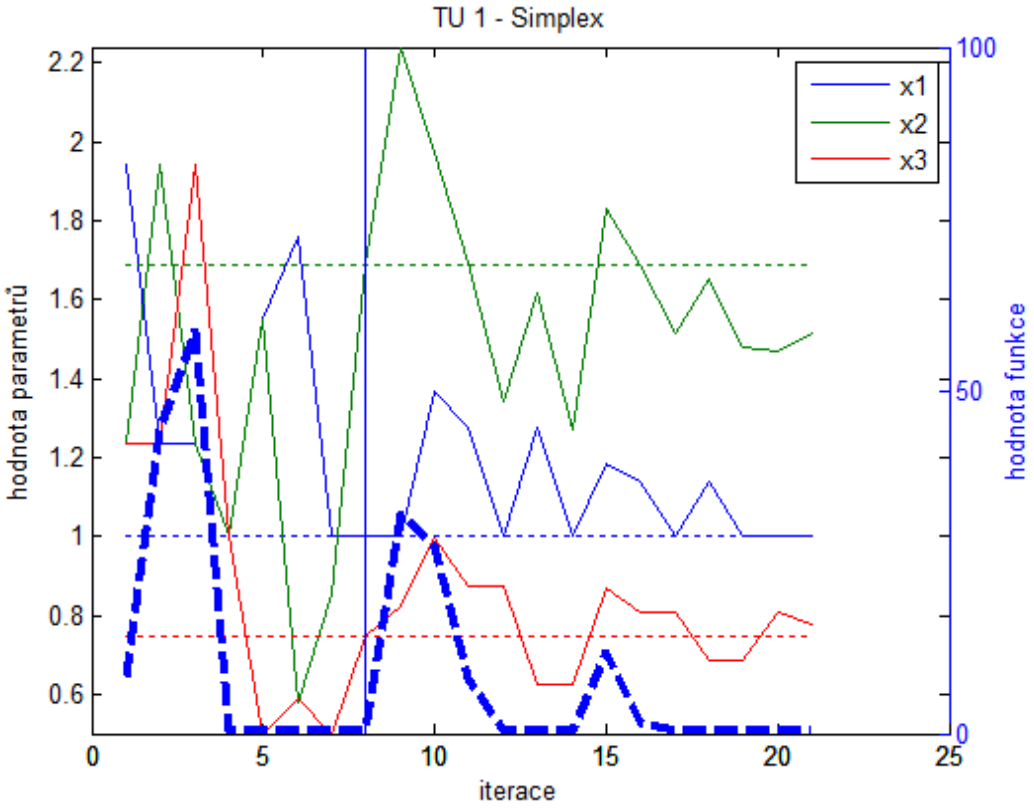


Obrázek 8.13: TU1 – Evoluční algoritmus Fobj = 4,46e-3 za 200 iterací.

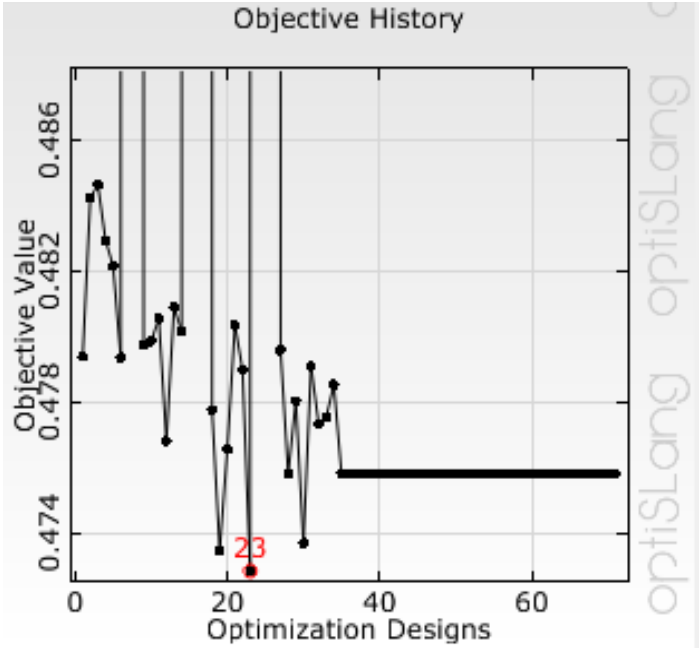


Obrázek 8.14: TU1 – Roj částic Fobj = 1,34e-5 za 394 iterací.

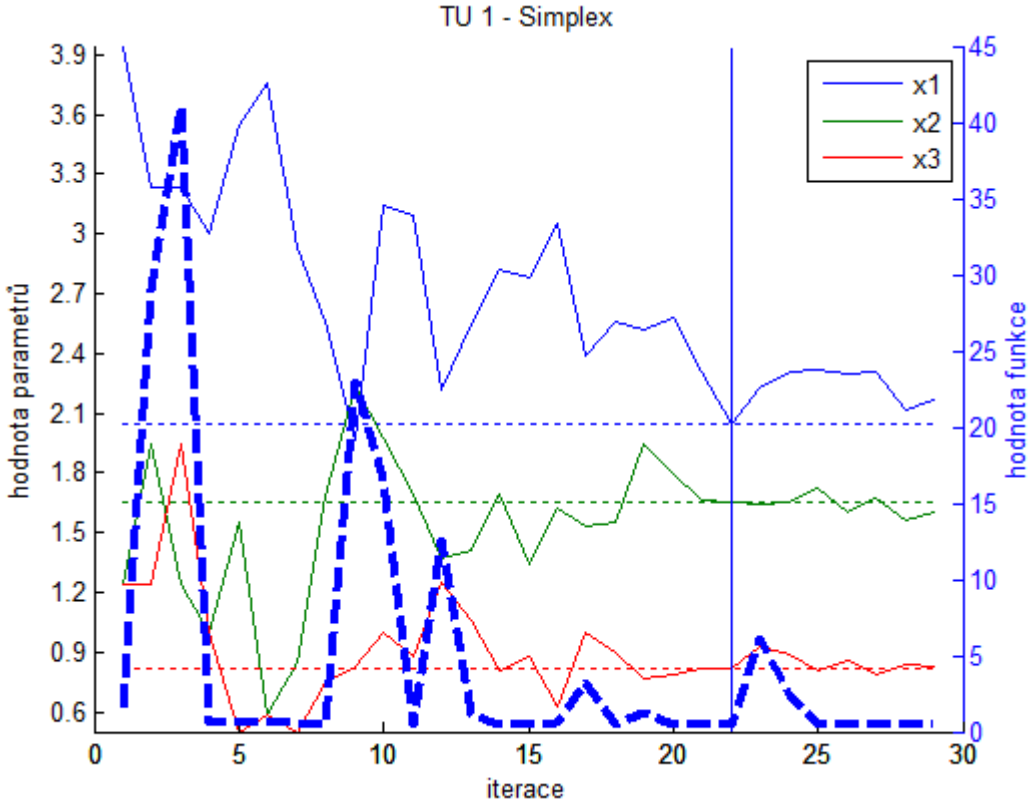
8.2.2. Průběh iterací testovací úlohy č.2



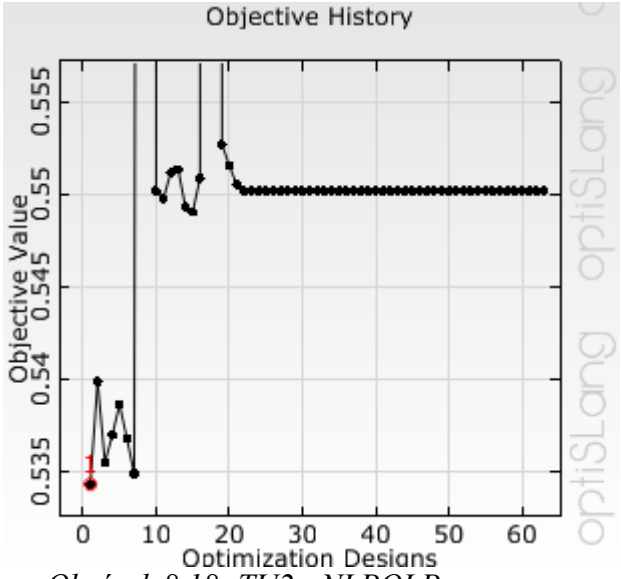
Obrázek 8.15: TU2 - simplex Nelder-Mead, Fobj = 4,77e-1 za 8 iterací.



Obrázek 8.16: TU2 - NLPQLP – optiSLang, Fobj = 4,73e-1 za 23 iterací.

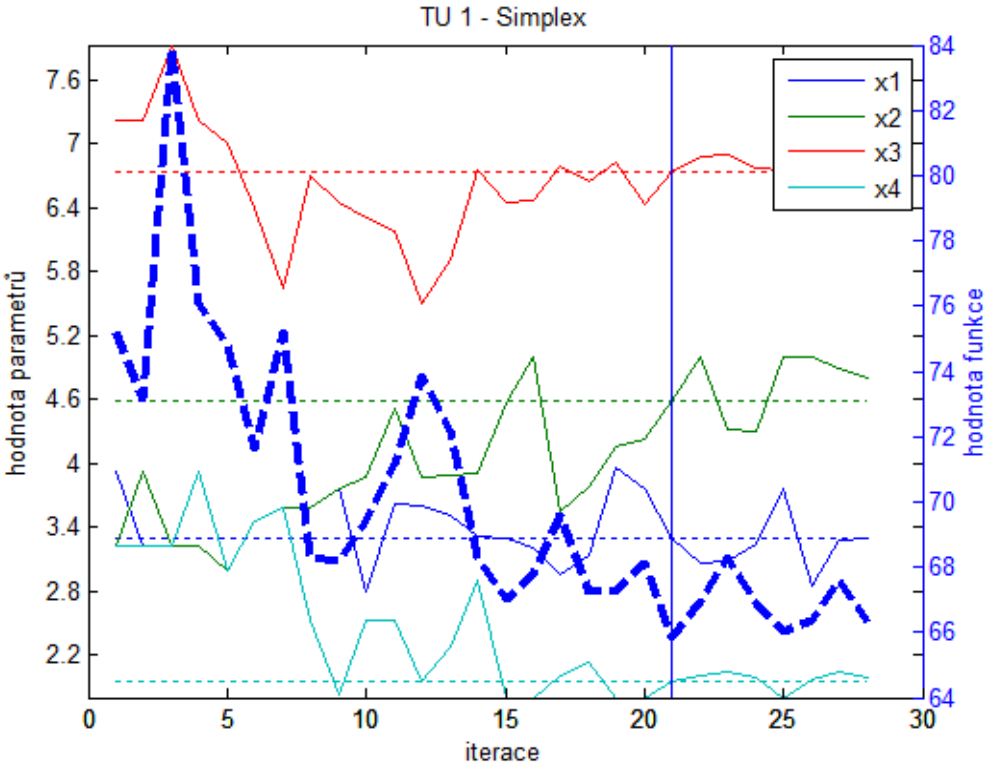


Obrázek 8.17: TU2 - simplex Nelder-Mead, přepočet pro bod [3 1 1]. Fobj = 4,83e-1 za 22 iterací z bodu [3 1 1] kde Fobj = 5,34e-1

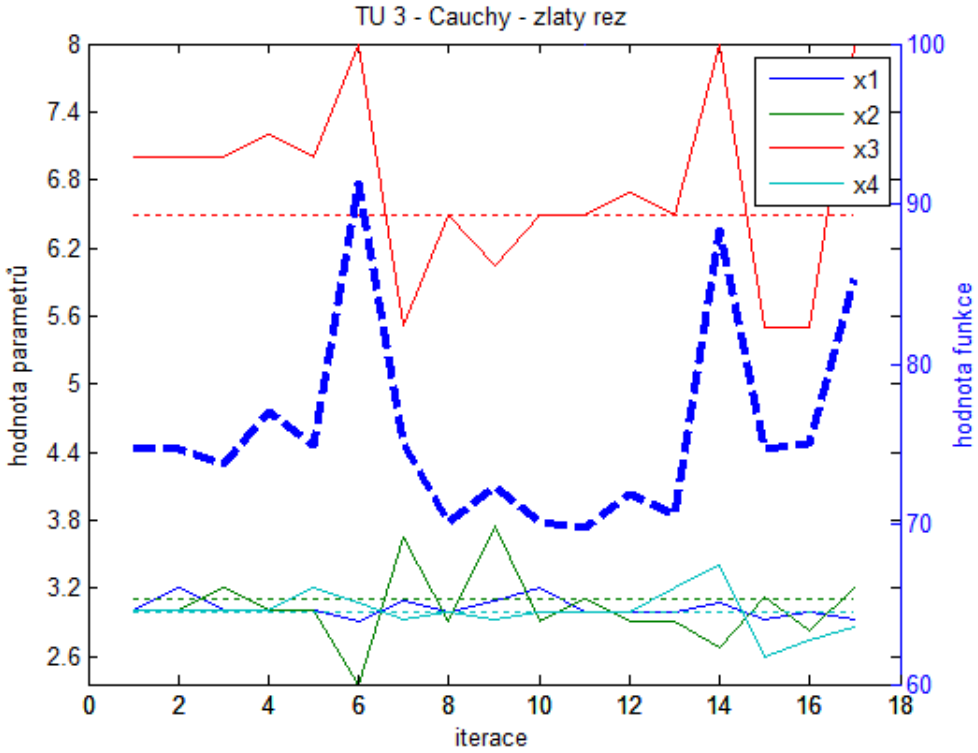


Obrázek 8.18: TU2 - NLPQLP, nekonvergující přepočet pro bod [3 1 1].

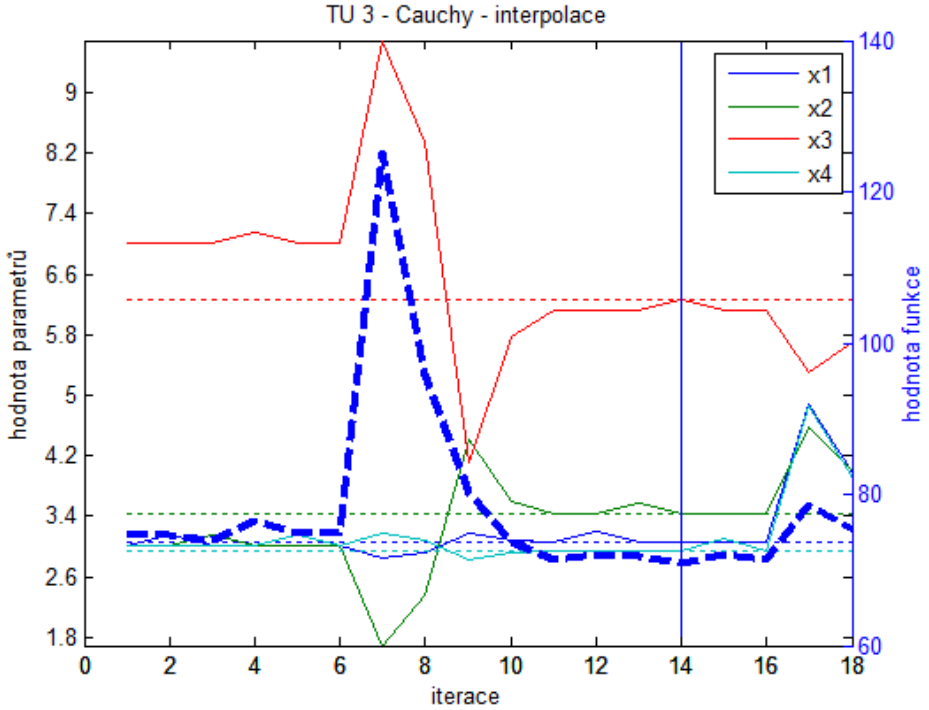
8.2.3. Průběh iterací testovací úlohy č.3



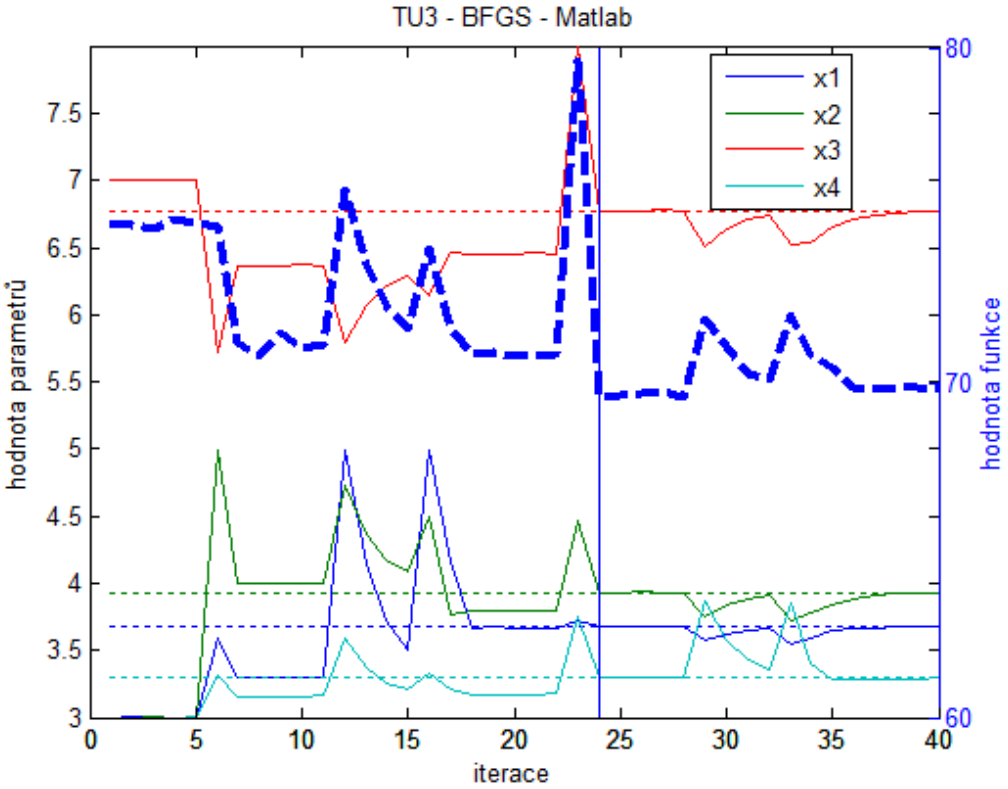
Obrázek 8.19: TU3 - simplex Nelder-Mead, Fobj = 65,82 za 21 iterací.



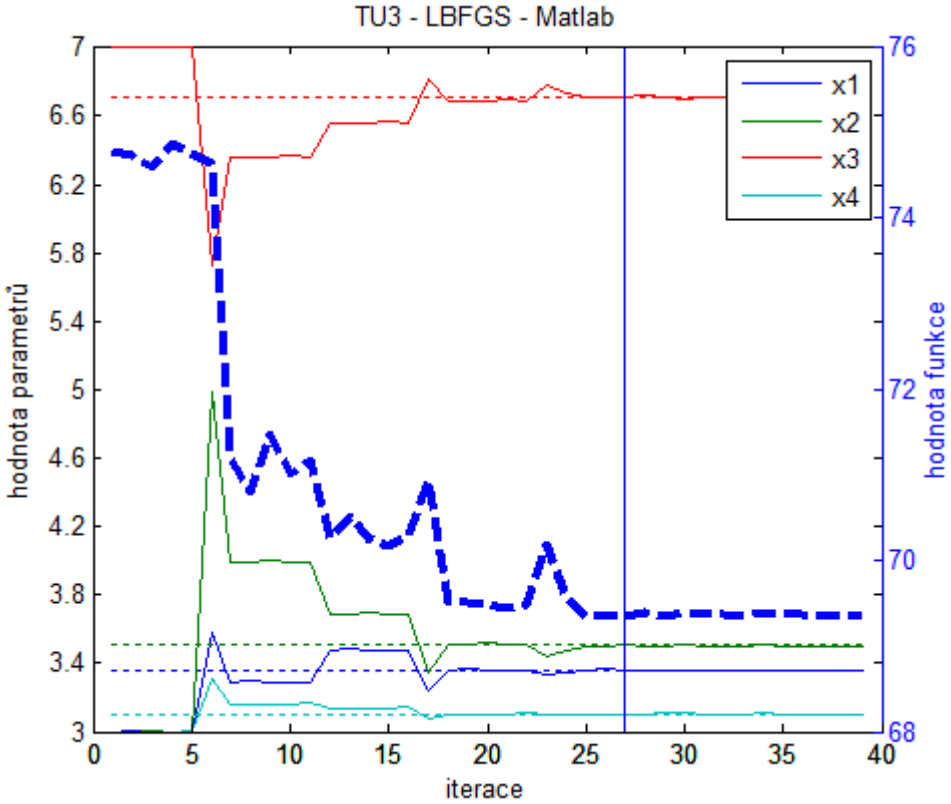
Obrázek 8.20: TU3 - Cauchy - Zlatý řez, Fobj = 70,07 za 11 iterací



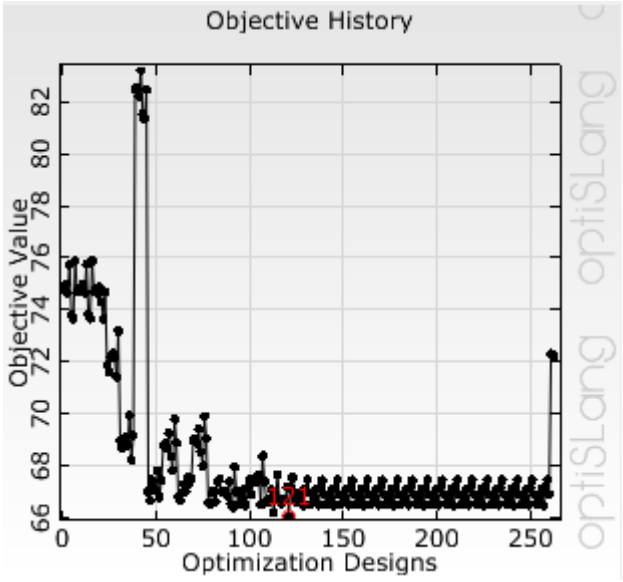
Obrázek 8.21: TU3 - Cauchy - interpolace, Fobj = 70,98 za 14 iterací (výpočet havaroval po 18ti iteracích z důvodu chyby v APDL souboru)



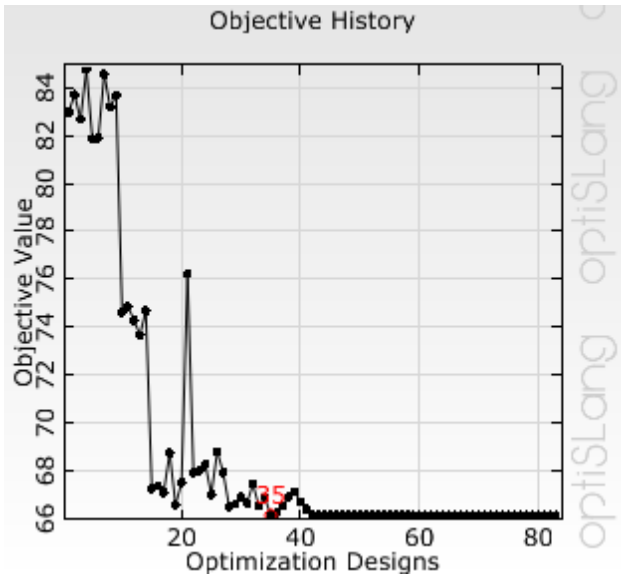
Obrázek 8.22: TU3 - BFGS - Matlab, Fobj = 69,57 za 24 iterací.



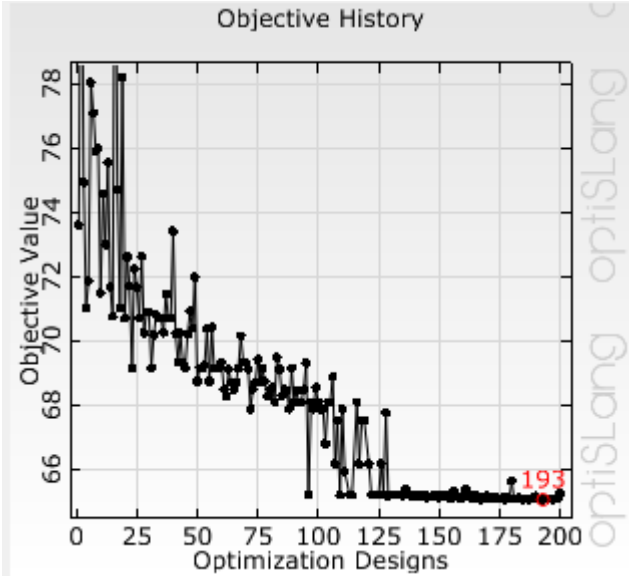
Obrázek 8.23: TU3 - LBFGS - Matlab, Fobj = 69,34 za 27 iterací.



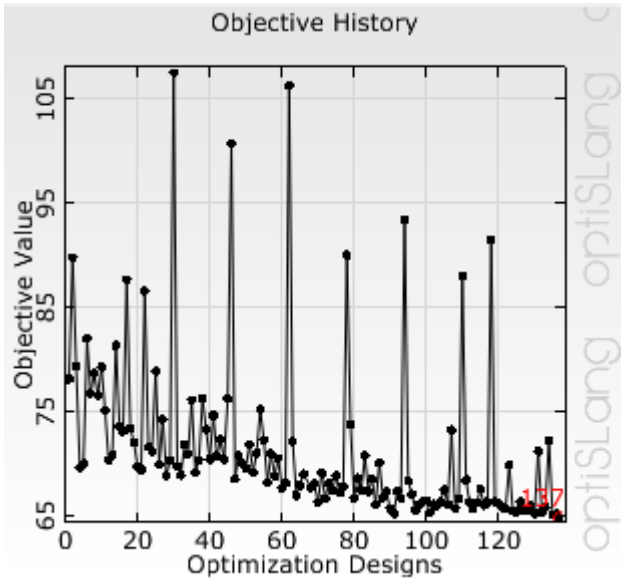
Obrázek 8.24: TU3 - LBFGS – optiSLang, Fobj = 66,06 za 121 iterací



Obrázek 8.25: TU3 - NLPQLP – optiSLang, Fobj = 6,61e-1 za 36 iterací



Obrázek 8.26: TU3 - Evoluční algoritmus, Fobj = 65,04 za 200 iterací



Obrázek 8.27: TU3 - Roj částic, Fobj = 64,75 za 137 iterací

8.3. Příloha 3 – seznam přiložených souborů na CD

Složka \Prilohy\APDL_Testing_Problems

Obsahuje základní APDL soubory nutné ke spuštění testovaných úloh programu ANSYS:

TU1-bcan3.mac

TU2-hrnalis.mac

TU3-vrub_make.mac

Složka \Prilohy\Matlab\ Algorithms

Soubory pro optimalizaci a vizualizaci v programu Matlab.

F.m – soubor spouštějící vyhodnocení funkce F(x), volá ANSYS v batchi s návratem hodnoty obj.fce jako output.txt

simp_opt.m – algoritmus Simplex Nelder-Mead

simpl_start.m – generování prvního simplexu pro simp_opt.m

order.m – seřazovací nástroj nezbytný pro simp_opt.m

CauchyGrad.m – algoritmus Cauchyho metody

ZlatyRezExpUru.m – m. zlatého řezu pro CauchyGrad.m

quadint.m – kvadratická interpolace pro CauchyGrad.m

pomocné:

nosnikanal.m – výpočet a vizualizace analytického výpočtu a vizualizace MKP výpočtu z bodů exportovaných souborem *TU1-bcan3.mac*

checklimits.m – zkontroluje zda-li je nový bod v zadaném intervalu a případně vrátí upravenou hodnotu proměnných

Složka \Prilohy\Matlab\ Design_exploration_4D_plot

Visualization_4d.m – nástroj na 4D vizualizaci, dostupné z webu: http://www.mathworks.com/matlabcentral/fileexchange/13503-4-dimensional-visualization

Generator.m – generuje matici, potřebnou pro 4D vizualizaci, v nastaveném rozsahu pro F(x)

Sliceomat.m – 4D vizualizace s iso povrchy

http://www.mathworks.com/matlabcentral/fileexchange/764

Složka \Prilohy\Matlab Optimalizace Matlab-ANSYS Workbench

Vyhodnocení F(x) pro ANSYS Workbench 13.0. Nebylo v této prácipoužito, ale může se hodit.

Složka \Prilohy\optiSLang

Soubory k řešení úlohy programu *optiSLang* včetně nastavení a výstupů všech provedených analýz.